

University of Groningen

Going Backward

Wassenaar, Tsjerk A.; Pluhackova, Kristyna; Böckmann, Rainer A.; Marrink, Siewert J.; Tieleman, D. Peter

Published in:
Journal of Chemical Theory and Computation

DOI:
[10.1021/ct400617g](https://doi.org/10.1021/ct400617g)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2014

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Wassenaar, T. A., Pluhackova, K., Böckmann, R. A., Marrink, S. J., & Tieleman, D. P. (2014). Going Backward: A Flexible Geometric Approach to Reverse Transformation from Coarse Grained to Atomistic Models. *Journal of Chemical Theory and Computation*, 10(2), 676-690. <https://doi.org/10.1021/ct400617g>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

```
#!/bin/bash

# This file contains the source code for the scripts and the mapping definitions.
# When executed it will
# - write initram.sh
# - write backward.py
# - make directory Mapping
# - write Mapping/_init_.py
# - write mapping files

# initram.sh
cat << __INITRAM_SH__ > initram.sh
#!/bin/bash

PROGRAM=initram.sh
VERSION=0.1
VERSTAG=devel-20130709-21-TAW
AUTHOR="Tsjerk A. Wassenaar, PhD"
YEAR="2013"
AFFILIATION="
University of Calgary
2500 University Drive NW
Calgary, Alberta
Canada, T2N 1N4"

CMD="\$0 \$@"

DESCRIPTION="\$(cat << __DESCRIPTION__
This is a convenience script to convert a coarse-grained system
into a united-atom or all-atom representation by projection
and subsequent relaxation. The script is a wrapper around
backward.py, which is called for the projection of the coarse-
grained system to unit-atom or all-atom, and GROMACS, which is
used for energy minimization and further relaxation using
molecular dynamics.

The script requires a COARSE-GRAINED input structure and a
corresponding ATOMISTIC topology file. The topology file will
be used to define the target particle list, allowing setting
protonation states, virtual sites, and even mutations by
providing an alternative topology. Atom list, atom names and
residue names from the topology file take precedence over the
ones in the structure file.

The script has a number of options to control the backmapping
process. The default options are chosen to give a robust protocol,
which can be used for backmapping of virtually any system. This
protocol is tested for systems containing protein, membrane and
solvent, with a total target size of up to a million particles.

The default protocol consists of the following steps:

1. Projection
2. Energy minimization (500 steps), excluding non-bonded interactions
   between atoms in selected groups, e.g. membrane and protein.
3. Energy minimization (500 steps)
4. Position restrained NVT simulation (300K), with time step 0.2 fs
5. Position restrained NVT simulation (300K), with time step 0.5 fs
6. Position restrained NVT simulation (300K), with time step 1.0 fs
7. Position restrained NVT simulation (300K), with time step 2.0 fs

These steps can be modified using the options, as listed below.

It is possible to extend the protocol using a number of user-supplied
run parameter (.mdp) files. These will be run after step 7. Multiple
mdp files can be supplied, which will be processed in the order given
(e.g., -mdp param1.mdp -mdp param2.mdp).

At the end of the process timing information is given summarizing the
time per step and the cumulative run time.

__DESCRIPTION__
)

# These will be looked for before running
DEPENDENCIES=(backward.py grompp mdrun)

# Get script directory
#
# A: Test if program name contains a slash (1)
# If so, change directory to program directory (2)
# B: Echo working directory, which will always be
# the program directory
#
# |-----A-----| |B|
# |-----1-----| |---2---|
SDIR="\$( [[ \$0 != \${0%/*} ]] && cd \${0%/*}; pwd )"

# - main options:
INP=
TOP=
OUT=backmapped.gro
OTF=backmapped.top
NDX=backmapped.ndx
RAW=projected.gro
BW=0-backward.gro
CG=martini
AA=gromos54a7
NUM=1
NP=0
KICK=0.05
KEEP=false
TRJ=false
EMSTEPS=500
NBSTEPS=500
MDSTEPS=500
DT=0.0002,0.0005,0.001,0.002
MDP=()
POSRE=true

# - em/md options

OPTIONS="\$(cat << __OPTIONS__

## OPTIONS ##

INITRAM options:
-f      Input coarse grained structure
-p      Input atomistic target topology
-po     Output processed topology
-o      Output atomistic structure
-from   Coarse grained force field
-to     Target forcefield
-n      Number of runs

*FILE: None
*FILE: None
*FILE: \${OTP}
*FILE: \${OUT}
*STR: \${SCG}
*STR: \${SAA}
*INT: \${NUM}
```

```

-np      Number of processors/threads          *INT:    \ $NP
-fc      Position restraint force constant      *FLOAT:  None
-kick    Random kick size                     *FLOAT:  \ $KICK
-keep    Do not delete intermediate files       *BOOL:   false
-trj     Write a trajectory for each stage of relaxation *BOOL:   false
-em      Number of steps for EM with bonded interactions only *INT:    \ $EMSTEPS
-nb      Number of steps for EM with nonbonded interactions *INT:    \ $NBSTEPS
-md      Number of steps for MD cycles          *INT:    \ $MDSTEPS
-dt      Time steps for successive MD cycles (comma separated list) *STR:    \ $DT
-nopr    Disable position restraints           *BOOL:   false
-mdp     User provided MDP file for post-hoc simulations *FILE:   None
         Multiple instances possible

__OPTIONS__
)

USAGE ()
{
    cat << __USAGE__

\ $PROGRAM version \ $VERSION:

\ $DESCRIPTION

\ $OPTIONS

(c) \ $YEAR \ $AUTHOR
\ $AFFILIATION

__USAGE__
}

BAD_OPTION ()
{
    echo
    echo "Unknown option \"$1\" found on command-line"
    echo "It may be a good idea to read the usage:"
    echo -e \ $USAGE

    exit 1
}

while [ -n "$1" ]; do
    case $1 in
        -h)          USAGE          ; exit 0 ;;
        # File options
        -f)          INP=\ $2        ; shift 2; continue ;;
        -p)          TOP=\ $2        ; shift 2; continue ;;
        -po)         OTP=\ $2        ; shift 2; continue ;;
        -o)          OUT=\ $2        ; shift 2; continue ;;
        -n)          NUM=\ $2        ; shift 2; continue ;;
        -np)         NP=\ $2         ; shift 2; continue ;;
        -from)       CG=\ $2         ; shift 2; continue ;;
        -to)         AA=\ $2         ; shift 2; continue ;;
        -kick)       KICK=\ $2       ; shift 2; continue ;;
        -keep)       KEEP=true       ; shift ; continue ;;
        -trj)        TRJ=true        ; shift ; continue ;;
        -em)         EMSTEPS=\ $2    ; shift 2; continue ;;
        -nb)         NBSTEPS=\ $2    ; shift 2; continue ;;
        -md)         MDSTEPS=\ $2    ; shift 2; continue ;;
        -dt)         DT=\ $2         ; shift 2; continue ;;
        -nopr)       POSRE=false     ; shift ; continue ;;
        -mdp)        MDP[\ $1#MDP[0]]=\ $2 ; shift 2; continue ;;
        *)          BAD_OPTION $1;;
    esac
done

cat << __RUNINFO__

\ $PROGRAM version \ $VERSION:

(c) \ $YEAR \ $AUTHOR
\ $AFFILIATION

Now executing...

\ $CMD

__RUNINFO__

# Bookkeeping

# Check input
[[ -z \ $INP ]] && echo FATAL ERROR: MISSING INPUT STRUCTURE FILE && exit 1
[[ -z \ $TOP ]] && echo FATAL ERROR: MISSING INPUT TOPOLOGY FILE && exit 1

# Check dependencies
missing=false
echo Checking dependencies:
for i in \ ${DEPENDENCIES[@]}
do
    echo -n "\ $i ... "
    which \ $i || which \ $SDIR/\ $i || missing=true
    \ $missing && echo Missing dependency: \ $i && exit 1
done

# Array for temporary files
GARBAGE=()

# Function for trashing temporary files
trash()
{
    for item in \ $@; do GARBAGE[\ ${#GARBAGE[@]}]=\ $item; done
}

# Define for position restraints
\ $POSRE && MDPDEF=-DPOSRES || MDPDEF=

# Starting time
START=\ $(date +%s)

#####
## STEP 1: GENERATING STARTING STRUCTURE ##
#####

GRO=\ $BW

B="\ $SDIR/backward.py -f \ $INP -raw \ $RAW -o \ $GRO -kick \ $KICK -sol -p \ $TOP -po \ $OTP -n \ $NDX -from \ $CG -to \ $AA"
echo \ $B; \ $B || exit

BM=\ $(date +%s)

```

```

trash \${RAW} \${GRO}

#####
## STEP 2: EM WITHOUT NONBONDED INTERACTIONS ##
#####

# Step counter
i=1

# Basename
BASE=\${i}-EM

# Run parameter file
mdp=\${BASE}.mdp

# If TRJ is true then write each frame
\${TRJ} && NSTXTCOUT=1 || NSTXTCOUT=0

cat << __MDP__ > \${mdp}

define=-DFLEXIBLE
integrator=steep
nsteps=\${EMSTEPS}
emstep=0.1
pbc=xyz

; Table extension is needed initially
table-extension=2

; During first steps nonbonded interactions
; are excluded within groups membrane and protein
energygrps=Protein Membrane Solvent
energygrp_excl=Protein Protein Membrane Membrane

; Usually no trajectory is written,
; but this can be changed (-trj)
nstxout=\${NSTXTCOUT}

__MDP__

# Set up for running
G="grompp -f \${mdp} -c \${GRO} -n \${NDX} -p \${OTP} -o \${BASE} -maxwarn 2"
echo \${G}; \${G} || exit

# Run
M="mdrun -deffnm \${BASE} -v -nt \${NP}"
echo \${M}; \${M}

# Timing
EM1=\$(date +%s)

# Mark files for deletion
trash \${BASE}.*

# Bookkeeping (increment run counter)
GRO=\${((i++))}-EM.gro

#####
## STEP 3: EM WITH NONBONDED INTERACTIONS ##
#####

# Basename for this cycle
BASE=\${i}-EM

# Turn all nonbonded interactions on and set the table_extension to default
# Change the number of steps
sed -e '/^nsteps/s/=.*/\${NBNSTEPS}/' -e '/^ *table-extension/s/^/;/' -e '/^ *energygrp_excl/s/^/;/' \${mdp} > \${BASE}.mdp
mdp=\${BASE}.mdp

# Set up for running
G="grompp -f \${mdp} -c \${GRO} -n \${NDX} -p \${OTP} -o \${BASE} -maxwarn 2"
echo \${G}; \${G} || exit

# Run
M="mdrun -deffnm \${BASE} -v -nt \${NP}"
echo \${M}; \${M}

# Timing
EM2=\$(date +%s)

# Mark files for deletion
trash \${BASE}.*

# Bookkeeping (increment run counter)
GRO=\${i}-EM.gro

#####
## STEP 4: MD WITH INCREASING TIME STEP ##
#####

# Array for collecting timing information
PRMD=()

# Set file tag
\${POSRE} && tag=mdpr || tag=mdnopr

# Unpack the list of time steps
ifs=\${IFS}
IFS=,
DT=(\${DT})
IFS=\${ifs}

for DELTA_T in \${DT[@]}
do
    BASE=\${((i++))}-\${tag}-\${DELTA_T}
    mdp=\${BASE}.mdp

cat << __MDP__ > \${mdp}
define                = \${MDPDEF}
integrator             = md
nsteps                = \${MDSTEPS}
dt                    = \${DELTA_T}
pbc                   = xyz

rcoulomb              = 0.9
rlist                 = 0.9
rvdw                  = 0.9

tcoupl                = v-rescale

```

```

ref_t           = 200
tau_t           = 0.1
tc_grps         = System

gen_vel         = yes
gen_temp        = 300

constraints      = all-bonds

nstxtcout       = \NSTXTCOUT

__MDP__

# Set up run
G="grompp -f \${mdp} -c \${GRO} -r \${BW} -p \${OTP} -o \${BASE} -maxwarn 2"
echo \${G}; \${G} || exit

# Perform run
M="mdrun -deffnm \${BASE} -v -nt \${NP}"
echo \${M}; \${M}

# Collect timing information
PRMD[\${#PRMD[@]}]=\$(date +%s)

# Collect garbage
trash \${BASE}.*

# Increment counter
GRO=\${BASE}.gro

done

#####
## STEP 5: MD WITH USER PROVIDED MDP FILES ##
#####

MD=()

for mdp in \${MDP[@]}
do

    m=\${mdp%.mdp}
    tag=\${(++i)}-md-\${m##*/}
    G="grompp -f \${mdp} -c \${GRO} -r \${BW} -p \${OTP} -o \${tag} -maxwarn 2"
    echo \${G}; \${G} || exit
    M="mdrun -deffnm \${tag} -v -nt \${NP}"
    echo \${M}; \${M}
    GRO=\${tag}.gro

    MD[\${#MD[@]}]=\$(date +%s)
done

# Copy last structure for output
cp \${GRO} \${OUT}

# Trash files if not keeping them
if ! \${KEEP}
then
    trash *mdout.mdp*

    echo Removing files:
    echo \${GARBAGE[@]}
    rm \${GARBAGE[@]}
fi

echo
echo "Timing (seconds):"
echo =====
printf "%-15s %5d %5d\n" Backmapping: \${((BM-START))} \${((BM-START))}
printf "%-15s %5d %5d\n" EM1: \${((EM1-BM))} \${((EM1-START))}
printf "%-15s %5d %5d\n" EM2: \${((EM2-EM1))} \${((EM2-START))}
P=\${EM2}
for ((i=0; i<\${#DT[@]}; i++))
do
    T=\${PRMD[\${i}]}
    printf "%-15s %5d %5d\n" PRMD-\${DT[\${i}]}: \${((T-P))} \${((T-START))}
    P=\${T}
done
for ((i=0; i<\${#MDP[@]}; i++))
do
    T=\${MD[\${i}]}
    printf "%-15s %5d %5d\n" MD-\${MDP[\${i}]}: \${((T-P))} \${((T-START))}
    P=\${T}
done
echo -----
printf "%-15s %5d\n\n\n" Total: \${((T-START))}

__INITRAM_SH__

# backward.py
cat << __BACKWARD_PY__ > backward.py
#!/usr/bin/env python

## Mapping from atomistic to coarse grained and vice versa

version="devel_130709.21_TAW"
authors=["Tsjerk A. Wassenaar"]

##

import sys, random, math, re, os, itertools
import Mapping

##

# Some definitions

AminoAcids      = "ALA CYS ASP GLU PHE GLY HIS ILE LYS LEU MET ASN PRO GLN ARG SER THR VAL TRP TYR ACE NH2".split()
protein_stuff   = list(AminoAcids)

NucleicAcids    = "C G A T U DC DG DA DT DCYT DGUA DADE DTHY CYT GUA ADE THY URA".split()
nucleic_stuff   = list(NucleicAcids)

Ions            = "NA NA+ CL CL-".split()

##

# I. Read structure
# II. Do mapping
# III. Write structure

# Force field levels

```

```

# Coarser force fields have higher rank
levels = {
    "martini": 2,
    "gromos": 1,
    "gromos43a2": 1,
    "gromos45a3": 1,
    "gromos53a6": 1,
    "gromos54a7": 1,
    "charmm": 0,
    "charmm27": 0,
    "charmm36": 0,
    "amber": 0,
    "amber94": 0,
    "amber96": 0,
    "amber99": 0,
    "amber99sb": 0,
    "amber03": 0,
    "amberGS": 0,
}

# Solvent and ions are a bit special. They usually map
# multiple molecules to a single bead. It seems best to
# map idealized configurations onto each bead.
fourWaters = [
    ("OW", -0.08,-0.08,-0.08),
    ("HW1", -0.08,-0.01,-0.01),
    ("HW2", -0.01,-0.01,-0.08),

    ("OW", -0.08, 0.08, 0.08),
    ("HW1", -0.01, 0.08, 0.01),
    ("HW2", -0.01, 0.01, 0.08),

    ("OW", 0.08, 0.08,-0.08),
    ("HW1", 0.08, 0.01,-0.01),
    ("HW2", 0.14, 0.14,-0.14),

    ("OW", 0.08,-0.08, 0.08),
    ("HW1", 0.01,-0.08, 0.01),
    ("HW2", 0.14,-0.14, 0.14),
]

solvent = {
    "SOL": [ ("OW", 0,0,0), ("HW1", 0.1,0,0), ("HW2", 0,0.1,0)],
    "W": fourWaters,
    "PW": fourWaters,
    "ION": [ ("ION",0.00,0.00,0.00)] + fourWaters,
    "CL": [ ("CL",0.00,0.00,0.00)] + fourWaters,
    "CL-": [ ("CL",0.00,0.00,0.00)] + fourWaters,
    "NA": [ ("NA",0.00,0.00,0.00)] + fourWaters,
    "NA+": [ ("NA",0.00,0.00,0.00)] + fourWaters,
}

solvent_stuff = solvent.keys()
ion_stuff = ["ION","CL","CL-","NA","NA+"]

# Number of residues mapping to/from bead
# This should list the residues that have mapping
# other than 1.
mapnum = {"SOL": 1, "W": 4, "PW": 4, "ION": 5, "CL": 5, "CL-": 5, "NA": 5, "NA+": 5}

#####
#####

### ||| WARNING: ||| ###
### ||| PRIVATE PARTS DOWN THERE ||| ###
### ||| EXPLICIT CONTENT ||| ###
### VVV VVV ###

#####
#####

def kick(x,u):
    return x+(random.random()-0.5)*u

#####
## STAGE 1: READ ATOMISTIC TOPOLOGY ##
#####

# Need to extract moleculetypes, residues and atom lists

# Set the pattern for matching files to be included
includePattern = re.compile('#include "(.*)"')

# Gromacs force field directory
gmxlib = os.environ.get("GMXLIB")
if not gmxlib:
    gmxdat = os.environ.get("GMXDATA")
    if gmxdat:
        gmxlib = os.path.join(gmxdat,"gromacs","top")
    else:
        gmxlib="."

# The following function finds and follows #included files
def reciter(filename):
    # Set the directory of the filename so we know where to expect #included files
    dir = os.path.dirname(filename)

    # Iterate over the lines
    for line in open(filename):

        # Check for an #include statement; yield the line if there is none
        if line.strip().startswith("#include"):

            # Extract the #include filename
            matches = re.findall(includePattern,line)

            if matches:
                fr = matches[0]

                if not os.path.exists(fr):
                    fr = os.path.join(dir,matches[0])

                if not os.path.exists(fr):
                    fr = os.path.join(gmxlib,matches[0])

                if not os.path.exists(fr):
                    yield "; " + line + " ; File not found\n"
                else:
                    for j in reciter(fr):
                        yield j
            else:

```



```

pdbline = "ATOM %5d %4s %4s %1s%4d    %8.3f%8.3f%8.3f%6.2f%6.2f\n"
return pdbline%(i,atom[0][:4],atom[1][:4],atom[3],atom[2],10*x,10*y,10*z,1,0)

def groAtom(a):
#012345678901234567890123456789012345678901234567890
iPRN      N      1      4.168    11.132    5.291
## ==> atom name, res name, res id, chain, x, y, z
return (str(a[10:15]), str(a[5:10]), int(a[:5]), " ", float(a[20:28]),float(a[28:36]),float(a[36:44]))

def get_calpha_xyz(r):
for i in r:
if i[0].strip() in ("CA","BB","BAS"):
return i[4:7]

def is_terminal(a, b, box, invbox):
return not a or not b or dist(a,b,box,invbox) > 0.7

class Structure:
def __init__(self,other,strict=False):

if type(other) == str:
lines = open(other).readlines()
else:
lines = other

# Try extracting PDB atom/hetatm definitions and set the box
self.box = None
rest = []
self.atoms = [pdbAtom(i,strict) for i in lines if isPDBAtom(i) or rest.append(i)]
if not self.atoms:
# This should be a GRO file - get the atom count
n = int(lines[1])+2
self.atoms = [groAtom(i) for i in lines[2:n]]
b = [float(i) for i in lines[n].split()] + 6*[0] # Padding for rectangular boxes
self.box = [[b[0],b[3],b[4]], [b[5],b[1],b[6]], [b[7],b[8],b[2]]] # Full definition xx,xy,xz,yx,yy,yz,zx,zy,zz
else:
# Make sure there is a box definition
b = [i for i in rest if i.startswith("CRYST1")]
if b:
self.box = pdbBoxRead(b[-1])

# Build a residue list
self.residues = [[self.atoms[0]]]
for i in self.atoms[1:]:
if i[1:4] != self.residues[-1][-1][1:4]:
self.residues.append([])
self.residues[-1].append(i)

# Extract the sequence
self.sequence = [ i[0][1].strip() for i in self.residues ]

# PBC handling
# To 'unbreak' residues, subtract the coordinates of the first atom
# convert to box coordinates and truncate. Convert back to Cartesian
# coordinates and add to the coordinates of the first atom.
A, B = None, None
if self.box:
A = zip(*self.box)
try:
B = m_inv(A)
self.residues = [ unbreak(i,A,B) for i in self.residues ]
except ZeroDivisionError:
print "Non-invertable box. Not able to unbreak molecules..."

# Check for protein chains and breaks
# List the coordinates for amino acid backbone
protein = [ i[0][1].strip() in AminoAcids and get_calpha_xyz(i) for i in self.residues ]
termini = [ is_terminal(i,j,A,B) for i,j in zip([False]+protein,protein+[False]) ]
self.nterm = [ j and i for i,j in zip(termini, protein) ]
self.cterm = [ j and i for i,j in zip(termini[1:],protein) ]

# Set chain backbones based on termini. Begin with a 'chain' unless the first residue is protein.
backbone = [[]]
for i,j,k in zip(self.nterm,self.terms,self.residues):
if i and backbone[-1]:
# We have a chain start, and the last chain is not empty: add a chain
backbone.append([])
# Try to fetch a C-alpha or backbone bead
backbone[-1].append(get_calpha_xyz(k))
if j:
# If this is a C terminus, add a new list
backbone.append([])

# Maybe we just added an empty list to the backbone, like if the last residue is a C-terminal
if backbone and not backbone[-1]:
backbone.pop()

# For each protein chain, positions are estimated for backbone atoms and set as a dictionary:
# {"N": (x,y,z), "CA": (x,y,z), ...}
self.backbone = []
for chain in backbone:
if not all(chain):
for i in chain:
self.backbone.append(False)
continue

# Set a dictionary for each residue. The dictionary will contain entries
# N, H, CA, HA, C, O
bb = [dict() for i in chain]

# Determine vector to each next residue
dl2 = [vsub(j,i) for i,j in zip(chain,chain[1:])]

# Determine the vector to each third residue
dl3 = [vsub(j,i) for i,j in zip(chain,chain[2:])]

# The crossproducts between actual and predicted vectors
# These end up corresponding surprisingly well to the backbone oxygen/hydrogen
# positions in both alpha-helix and beta-sheet.
# Only turns appear to have inverted peptide planes... Strange.
crsp = [normalize(crossprod(a,p)) for a,p in zip(dl2,dl3)]

# Copy the last direction vector to set C/O on the last residue
dl2.append(dl2[-1])
crsp.append(crsp[-1])
crsp.append(crsp[-1])

```



```

# For the first N/H we use the direction towards the next residue
px,py,pz = d12[0]
qx,qy,qz = crsp[0]
for i in range(len(bb)):
    x, y, z = chain[i]
    dx,dy,dz = d12[i]
    cx,cy,cz = crsp[i]

    # The coordinate stored for the residue was the CA/BB one
    bb[i]["CA"] = (x,y,z)

    # C/O are about one third towards the next CA
    # The are shifted in the direction of the d12/d13 crossproduct
    bb[i]["C"] = (x+dx/3+0.035*cx, y+dy/3+0.035*cy, z+dz/3+0.035*cz)
    bb[i]["O"] = (x+dx/3+0.155*cx, y+dy/3+0.155*cy, z+dz/3+0.155*cz)

    # N/H are about one third towards the previous CA
    # The are shifted in the direction opposite from the previous
    # d12/d13 cross product
    bb[i]["N"] = (x-px/3-0.035*qx, y-py/3-0.035*qy, z-pz/3-0.035*qz)
    bb[i]["H"] = (x-px/3-0.155*qx, y-py/3-0.155*qy, z-pz/3-0.155*qz)
    bb[i]["HN"] = bb[i]["H"]

    # Store the d12 direction vector and d12/d13 crossproduct
    px,py,pz = dx, dy, dz
    qx,qy,qz = cx, cy, cz

# Add the residue dictionaries to the backbone list
self.backbone.extend(bb)

def groBoxString(self):
    groBoxLine = "%10.5f%10.5f%10.5f%10.5f%10.5f%10.5f%10.5f%10.5f"
    if self.box:
        return groBoxLine % (self.box[0][0],self.box[1][1],self.box[2][2],
                               self.box[0][1],self.box[0][2],self.box[1][0],
                               self.box[1][2],self.box[2][0],self.box[2][1])
    else:
        return groBoxLine % (0,0,0,0,0,0,0,0)

class Topology:
    def __init__(self,other,out=None):
        # Process the topology file extract moleculetypes, atom lists and the molecule list
        self.molecules = []
        self.top = []

        # Processed topology
        # This is equal to the input target topology, but with all #include statements resolved
        if out:
            out = open(out,"w")

        # List of line numbers at which to find moleculetype definitions
        mols = []

        # Moleculetypes
        moltypes = []

        # Atoms per moleculetype
        atoms = []

        # Gromacs topology directive
        tag = re.compile('^ *\[ *(.*) *\]')
        # Last directive read (current)
        cur = None
        # Set line counter
        counter = 0
        # Iterate over lines, processing #included files
        for line in reciter(other):
            if out:
                out.write(line)

            # Increment the line counter
            counter += 1
            # Add the line to the (processed) topology
            self.top.append(line)
            # Strip leading and trailing spaces
            s = line.strip()
            # Lines starting with [ indicate a directive
            if s.startswith("["):
                # Extract the directive name
                cur = re.findall(tag,s)[0].strip()
                continue
            # Conditionals :S
            # Conditionals are simply skipped
            if s.startswith("#"):
                continue
            # Strip comments
            s = s.split('#')[0].strip()
            # Skip empty lines
            if not s:
                continue
            if cur == "moleculetype":
                moltypes.append(s.split()[0])
                atoms.append([])
                mols.append(counter)
                continue
            if cur == "system":
                mols.append(counter)
                continue
            if cur == "atoms":
                # Comments are already skipped
                a = s.split()
                atoms[-1].append((a[4],a[3],a[2],""))
                continue
            if cur == "molecules":
                # Each molecules entry has a moleculetype name and a number
                # The moleculetype name is added to the molecules list
                # as many times as the number indicates. This makes it easy
                # to expand the molecules to atoms.
                m = s.split()
                for j in range(int(m[1])):
                    self.molecules.append(m[0])

        if out:
            out.close()

        # Convert moleculetypes to dictionary
        self.moleculetypes = dict(zip(moltypes,atoms))

        molecules = [(i,len(list(j))) for i,j in itertools.groupby(self.molecules)]

        # Build a full atom list
        # The chain identifier is unique for each molecule
        # The moleculetype name is added as last element
        mr = zip(self.molecules, range(len(self.molecules)))

```

```

self.atoms = [[a,r,i,c,0,0,0,t] for t,c in mr for a,r,i,m in self.molculetypes[t]]

# Build a residue list
if self.atoms:
    self.residues = [[self.atoms[0]]]
    for i in self.atoms[1:]:
        if i[1:4] != self.residues[-1][-1][1:4]:
            self.residues.append([])
            self.residues[-1].append(i)
    else:
        self.residues = []

#####

## PARSING COMMAND LINE ARGUMENTS ##

# Very simple option class
class Option:
    def __init__(self,func=str,num=1,default=None,description=""):
        self.func = func
        self.num = num
        self.value = default
        self.description = description
    def nonzero(self):
        if self.func == bool:
            return self.value != None
        return bool(self.value)
    def __str__(self):
        return self.value and str(self.value) or ""
    def setvalue(self,v):
        if len(v) == 1:
            self.value = self.func(v[0])
        else:
            self.value = [ self.func(i) for i in v ]

# Description
desc = ""

# Option list
options = [
# option      type      number      default      description
    ("f",      Option(str,      1,      None, "Input GRO/PDB structure")),
    ("o",      Option(str,      1,      None, "Output GRO/PDB structure")),
    ("raw",    Option(str,      1,      None, "Projected structure before geometric modifications")),
#    ("c",      Option(str,      1,      None, "Output GRO/PDB structure of expanded CG beads for position restraints")),
    ("n",      Option(str,      1,      None, "Output NDX index file with default groups")),
    ("p",      Option(str,      1,      None, "Atomistic target topology")),
    ("po",     Option(str,      1,      None, "Output target topology with matching molecules list")),
    ("pp",     Option(str,      1,      None, "Processed target topology, with resolved #includes")),
    ("-atomlist", Option(str,      1,      None, "Atomlist according to target topology")),
    ("-fc",    Option(float,    1,      200, "Position restraint force constant")),
    ("-to",    Option(str,      1,      None, "Output force field")),
    ("-from",  Option(str,      1,      None, "Input force field")),
    ("-strict", Option(bool,    0,      None, "Use strict format for PDB files")),
    ("-nt",    Option(bool,    0,      None, "Use neutral termini for proteins")),
    ("-sol",   Option(bool,    0,      None, "Write water")),
    ("-kick",  Option(float,    1,      0, "Random kick added to output atom positions")),
]

# Parsing arguments
args = sys.argv[1:]
if '-h' in args or '--help' in args:
    print "\n",__file__
    print desc or "\nSomeone ought to write a description for this script...\n"
    for thing in options:
        print type(thing) != str and "%10s  %s"%(thing[0],thing[1].description) or thing
    print
    sys.exit()

# Convert the option list to a dictionary, discarding all comments
options = dict([i for i in options if not type(i) == str])

# Process the command line - list the options that were given
opts = []
while args:
    opts.append(args.pop(0))
    options[opts[-1]].setvalue([args.pop(0) for i in range(options[opts[-1]].num)])

## DONE PARSING ARGUMENTS ##

#####

## MAPPING ##

#### A. If a target topology was provided, read it in

top = options["-p"] and Topology(options["-p"].value,out=options["-pp"].value)

#### B. Read in the structure

struc = Structure(options["-f"].value,strict=options["-strict"].value)

#### C. Set the mapping dictionary

# Convert force field tags to lower case
# Default is backmapping from MARTINI to GROMOS53A6
# If to_ff == martini, default from ff = gromos
to_ff = options["-to"] and options["-to"].value.lower() or "gromos"
if to_ff == "martini" and not options["-from"]:
    from_ff = "gromos"
else:
    from_ff = options["-from"] and options["-from"].value.lower() or "martini"
mapping = Mapping.get(source=from_ff,target=to_ff)
backmapping = levels[from_ff] > levels[to_ff]
reslist = mapping.keys()

#### D. Iterate over atoms to write out, based on residue names

# Copy the residue list from the target topology
# This gives a list we can pop from, while keeping
# the original.
# The solvent residues are skipped.
topresidues = None
if top:

```

```

topresidues = [i for i in top.residues]
if options["-atomlist"]:
    atm = open(options["-atomlist"].value, "w")
    topatm = {}
    for i in topresidues:
        for j in i:
            atm.write("%s\n" % (i[j], j))
    atm.close()

# Iterate over residues
# If we are backmapping, we store the BB bead
# positions, to generate a spline, which we use
# afterwards to place the backbone atoms.
# To set the positions, we use some bookkeeping
# tricks for the atoms to place on, or relative
# to, the spline.
# The backbone list will end up being equal in
# length to the number of (amino acid) residues.
# It is processed afterwards to be three times
# the length. Indices are used to indicate which
# entry from the resulting interpolated spline
# list need to be taken for the position, and an
# offset (tuple) is added to control the placement
# of hydrogens and oxygens to N/C.
counter = 0
out = []
cg = []
raw = []
sol = []
ions = []
for residue, bb, nterm, cterm in zip(struct.residues, struct.backbone, struct.nterm, struct.cterm):

    counter += 1

    # Ignore solvent molecules from the topology
    while topresidues and topresidues[0][0][1] in solvent_stuff:
        topresidues.pop(0)

    # Unpack first atom
    first, resn, resi, chain, x, y, z = residue[0]

    # Extract residue name and atom list
    resn = resn.strip()
    atoms = [i[0].strip() for i in residue]

    # Just read one residue from the CG structure
    # If we have a topology, we need to check whether
    # the residue we just read matches the next in the
    # topology. Several cases are possible:
    #
    # - The residuenamename is equal in both cases:
    #   This is too easy! Just proceed and thank your deity.
    #
    # - The residues do not match, but the CG residuenamename
    #   matches the AA moleculetype name:
    #   This may happen with lipids, if the atomistic structure
    #   is split in residues like in the De Vries model.
    #   In this case, all residues corresponding to the molecule
    #   need to be read from the topology, based on the chain
    #   identifier.
    #
    # - The residuenamename does not match, but the residue does:
    #   The residues should match, or at least the first
    #   characters.
    #
    # - The residue does not match with either the residue or
    #   moleculetype from the atomistic topology:
    #   If the residue is solvent, then we leave the topology
    #   untouched, and the atoms are generated based on the
    #   mapping.

    # Check for solvent
    if resn in solvent.keys():
        cx, cy, cz = residue[0][4:7]
        for atom, x, y, z in solvent[resn]:
            # Should add random rotation
            if atom in ion_stuff:
                atom = atoms[0]
                ions.append((atom, resn, counter, chain, cx+x, cy+y, cz+z))
                # They are added at the end, which is safe, as the
                # ion position is taken from the CG bead position
                # anyway.
            else:
                # If we do not want solvent written then this is
                # a good time to break: the first atom of a stretch
                # of solvent. Note that this ensures that we write
                # ions if we have those.
                if not options["-sol"]:
                    break
                resn = "SOL"
                # Increase the counter if we have an oxygen.
                # A little hack to keep track of water molecules
                if atom[0] == "O":
                    counter += 1
                sol.append((atom, resn, counter, chain, cx+x, cy+y, cz+z))
        # Go to next residue
        continue

    # Read a residue from the target topology if we have one
    # Read several if the mapping so requires
    # Make an atom list from the residues read
    if topresidues:
        # Check whether the CG residue corresponds to the next AA residue
        # or to the next moleculetype
        topres = topresidues.pop(0)
        if resn != topres[0][1] and resn == topres[0][7]:
            # Add residues based on chain id
            while topresidues and topresidues[0][0][3] == topres[0][3]:
                topresidues.pop(0)
            topres = [i for j in range(mapnum.get(resn, 1)) for i in topresidues]
            # Set the residue name to the moleculetype name
            topres[0][3] = topres[0][7]
            target = zip(*topres)[0]
            # Check for duplicate atom names
            if not len(target) == len(set(target)):
                print "The target list for residue %s contains duplicate names. Relying on mapping file." % resn
                target = None
        else:
            target = None

    # Except for solvent, the residue name from a topology
    # takes precedence over the one from the structure.
    if target and topres[0][1] in mapping.keys():

```

```

resn = topres[0][1]

# Check if the residue is in the list
# or whether we have an ambiguity.
# In that case the first part of the
# residue proper is equal to what we have
# and the atom lists should be equal
if not resn in reslist:
    p = set(atoms)
    for i in reslist:
        if i.startswith(resn):
            if p == set([k for j in mapping[i].map.values() for k in j]):
                resn = i
                break

if not resn in mapping.keys():
    # If the residue is still not in the mapping list
    # then there is no other choice that to bail out
    raise ValueError, "Unknown residue: %s\n"%resn

o, r = mapping[resn].do(residue,target,bb,nterm,cterm,options["-nt"])
out.extend(o)
raw.extend(r)

## Write out

# Combine things

out.extend(sol)
out.extend(ions)
raw.extend(sol)
raw.extend(ions)

# Write out

if options["-o"]:
    dev = open(options["-o"].value,"w")
else:
    dev = sys.stdout

# Title
if backmapping:
    dev.write("Backmapped structure from MARTINI to %s\n"%options["-to"].value)
else:
    dev.write("Mapped structure from %s to MARTINI\n"%options["-from"].value)

# Atom count
dev.write("%5d\n"%len(out))

u = options["-kick"].value

# Atoms
idx = 1
for atom in out:
    # Regular atom
    nam,res,id,chn,x,y,z = atom
    if False and res not in solvent_stuff:
        x,y,z = kick(x,u),kick(y,u),kick(z,u)
    dev.write("%5d%-5s%-5s%-5d%8.3f%8.3f%8.3f\n"%(id%1e5,res,nam,idx%1e5,x,y,z))
    idx += 1

# Box
dev.write(struc.groBoxString()+"\n")

# Close if we were writing to file
if options["-o"]:
    dev.close()

# Write the "raw" structure obtained by projection
if options["-raw"]:
    dev = open(options["-raw"].value,"w")
    dev.write("Projected structure before modifications\n")
    dev.write("%5d\n"%len(raw))
    idx = 1
    for nam,res,id,chn,x,y,z in raw:
        dev.write("%5d%-5s%-5s%-5d%8.3f%8.3f%8.3f\n"%(id%1e5,res,nam,idx%1e5,x,y,z))
    idx += 1
    dev.write(struc.groBoxString()+"\n")

## Write the output topology
if options["-p"] and options["-po"]:
    po = open(options["-po"].value,"w")
    mol = False

    # Write everything up to the [ molecules ] directive
    # After that, write only lines which are not solvent or ions
    for i in open(options["-p"].value):
        s = i.strip()
        if "molecules" in i:
            # Make sure we are not dealing with a comment
            if s.startswith('#') and s[1:].strip().startswith("molecules"):
                mol = True
            # Skip empty lines and comments

            # Now skip the lines listing solvent and ion molecules
            if mol:
                if not s:
                    continue
                if s[0] != ":" and i.split()[0] in solvent_stuff:
                    continue

            po.write(i)

    # Add lines for solvent and ions
    sol = [(i[1],i[2]) for i in sol]
    sol = [a[0] for a,b in itertools.groupby(sol)]
    po.writelines(["%s %5d\n"%(a,len(list(b))) for a,b in itertools.groupby(sol)])

    ions = [(i[0],i[2]) for i in ions]
    ions = [a[0] for a,b in itertools.groupby(ions)]
    po.writelines(["%s %5d\n"%(a.replace("+","").replace("-",""),len(list(b)))
                    for a,b in itertools.groupby(ions)])

## Write an index file
if options["-n"]:
    # Index groups
    ndx_protein = []
    ndx_membrane = []
    ndx_solvent = []

```

```

for i,j in zip(range(1,1+len(out)),out):
    if j[1] in protein_stuff:
        ndx_protein.append(i)
    elif j[1] in solvent_stuff:
        ndx_solvent.append(i)
    else:
        ndx_membrane.append(i)

    ndx = open(options["-n"].value,"w")
    ndx.write("[ Protein ]\n"+"\n".join([str(i) for i in ndx_protein])+"\n")
    ndx.write("[ Membrane ]\n"+"\n".join([str(i) for i in ndx_membrane])+"\n")
    ndx.write("[ Solvent ]\n"+"\n".join([str(i) for i in ndx_solvent])+"\n")

__BACKWARD_PY__

# Create directory and descend
mkdir Mapping
cd Mapping

# Mapping/__init__.py
cat << __INIT_PY__ > __init__.py

import glob,os,re,math,sys,random

# Should version this... 130502-11 TAW
# More extensive support for geometric operations

# In the context of this module, a residue is a list of atoms, where each atom/item
# is a list or tuple with at least 7 values:
#
# (atom name (str), residue (str), residue id (int), chain (char), x (float), y (float), z (float))

# Crude mass for weighted averages. No consideration of united atoms.
# This will probably give only minor deviations, while also giving less headache
# We add B with a mass of 32, so BB and SC* will have equal weights
_mass = {'H': 1,'C': 12,'N': 14,'O': 16,'S': 32,'P': 31,'M': 0, 'B': 32}

# Normalization factor for geometric modifiers (nanometer)
_normfac = 0.125

# Listing of aminoacids
_ aminoacids = [
    "ALA", "CYS", "ASP", "GLU", "PHE", "GLY", "HIS", "ILE", "LYS", "LEU",
    "MET", "ASN", "PRO", "GLN", "ARG", "SER", "THR", "VAL", "TRP", "TYR",
    "ACE", "NH2",
]

# Determine average position for a set of atoms
def _average(a):
    if a:
        # Get some massy number for each atom
        # The masses are tailored for atomistic models.
        # For coarse-grained force fields it is usually
        # sufficient to have equal masses
        mxyz = [(i._mass.get(i[0][0],1),i[4],i[5],i[6]) for i in a if i] # Masses and coordinates
        mw = [sum(i) for i in zip([(m*x,m*y,m*z,m) for m,x,y,z in mxyz])] # Sums if weighted coordinates
        return [i/mw[3] for i in mw] # Centre of mass
    return None

def _vsub(a,b):
    return [i-j for i,j in zip(a,b)]

def _vadd(a,b):
    return [i+j for i,j in zip(a,b)]

def _normalize(a):
    l = math.sqrt(sum([i*i for i in a]))
    return [i/l for i in a]

def _crossprod(a,b):
    return [a[1]*b[2]-a[2]*b[1],a[2]*b[0]-a[0]*b[2],a[0]*b[1]-a[1]*b[0]]

def _r(a,kick):
    return a+random.random()*kick-kick/2

class ResidueMap:
    def __init__(self,target=None,source=None,atoms=None,mod=[],name=""):
        if atoms:
            # Setting mapping from an atom list
            # Format is:
            # number, aa, cg beads
            x = [i[1] for i in atoms]
            y = [i[2] for i in atoms]
            # For those atoms for which no source list is given
            # set the source equal to that of the previous one
            for i in range(len(y)):
                if not y[i]:
                    y[i] = y[i-1]

        if source:
            y = source

        if target:
            if not atoms:
                x = target

            assert len(x) == len(y)

            # Case of forward mapping: atomistic to martini
            # Initialize dictionary
            d = dict(zip(target,[[i for i in target]]))

            # Fill entries
            # The mapping is specified in full both ways, which
            # means that, e.g. for Martini, the result may differ
            # from the original mapping definition. This should
            # add stability, and is required to allow the double
            # mappings used in Martini for some residues.
            for u,v in zip(y,x):
                for j in u:
                    d[j].append(v)

            self.atoms = target

```

```

        self.map = d
    else:
        self.atoms = x
        self.map = dict(zip(x,y))

# Special cases
self.mod = mod

def do(self, residue, target=None, coords=False, nterm=False, cterm=False, nt=False, kick=0.05):
    # Given a set of source atoms with coordinates
    # return the corresponding list of mapped atoms
    # with suitable starting coordinates.

    # If a target list is given, match every atom against
    # the atoms in the ResidueMap. If an atom is not in
    # the definition, then it is returned with the
    # coordinates of the last atom that was in the list.

    # For amino acids, nterm/cterm will cause extra hydrogens/
    # oxygen to be added at the start/end of the residue.
    # If nt (neutral termini) is true, two hydrogens will be
    # added in stead of three if nterm is true and an additional
    # hydrogen will be added at the end if cterm is true.

    # Unpack first atom
    first, resn, resi, chain, x, y, z = residue[0]
    resn = resn.strip()

    # Check whether a target was supplied
    set_termini = not target

    # Target atoms list
    if target:
        # A target atom list can be given as a list of names
        # or as a list of atoms. In the latter case, each element
        # will be a list or tuple and we extract the names.
        if type(target[0]) in (list,tuple):
            target = [i[0].strip() for i in target]
        elif type(target) == str:
            target = target.split()
        else:
            # Make a copy to leave the original list untouched
            target = list(target)
    else:
        target = list(self.atoms)

    # Atoms we have; the source dictionary
    atoms = [ i[0].strip() for i in residue ]
    have = dict(zip(atoms, residue))

    # Set array for output; residue to return
    out = []

    # The target list is leading. Make sure that the atom list matches.
    # So, the actual atom list is built from the target list:
    atomlist = [i for i in target if i in self.atoms]

    # Go over the target particles; the particles we want
    # If we have a target topology, then there may be
    # additional particles to want, especially hydrogens.
    # These will be assigned to the previous heavy atom
    # from the want list.
    for want in atomlist:

        # If we have a target list, the atom will be in
        # (it would have been skipped otherwise), and we
        # can read up to the next we want.
        if coords:
            got = coords.get(want, _average([ have.get(i) for i in self.map[want] ]))
        else:
            got = _average([ have.get(i) for i in self.map[want] ])

        if not got:
            print "Problem determining mapping coordinates for atom %s of residue %s."%(target[0],resn)
            print "atomlist:", atomlist
            print "want:", want, self.map[want]
            print "have:", have.keys()
            print "Bailing out..."
            print target
            sys.exit(1)

    # This logic reads the atom we want together with all atoms
    # that are in the target list, but not in the residue
    # definition in the mapping dictionary, up to the next atom
    # that is in that definition.
    while target and (target[0] == want or target[0] not in self.atoms):
        name = target.pop(0)

        out.append((name,resn,resi,chain,got[0],got[1],got[2]))

        # If we have a target list given as argument to the function
        # then we ignore whatever is given for nterm/cterm.
        # Otherwise, the N-terminal/C-terminal additions are made
        # right after the (N)H/(C)O, if nterm/cterm are set
        if resn in _aminoacids and set_termini:

            # If this is an N-terminal amino acid, we may have
            # to add one or two hydrogens.
            if nterm:
                if resn in ("PRO","HYP") and name == "N":
                    if nt:
                        # Add one
                        out.append(("H",resn,resi,chain,got[0],got[1],got[2]))
                    else:
                        # Add two
                        out.append(("H1",resn,resi,chain,got[0],got[1],got[2]))
                        out.append(("H2",resn,resi,chain,got[0],got[1],got[2]))
                elif want == "H":
                    if nt:
                        # Add one
                        out.append(("H2",resn,resi,chain,got[0],got[1],got[2]))
                    else:
                        # Add two
                        out.append(("H2",resn,resi,chain,got[0],got[1],got[2]))
                        out.append(("H3",resn,resi,chain,got[0],got[1],got[2]))

            if cterm and want == "O":
                out.append((want,resn,resi,chain,got[0],got[1],got[2]))
            if nt:
                out.append(("H",resn,resi,chain,got[0],got[1],got[2]))

```

```
# Now add a random kick whenever needed to ensure that atoms overlap
for i in range(len(out)):
    for j in range(i):
        if out[i][4] == out[j][4] and out[i][5] == out[j][5] and out[i][6] == out[j][6]:
            # Equal coordinates: need fix
            x,y,z = out[i][4:7]
            out[i] = out[i][:4]+(_r(x,kick),_r(y,kick),_r(z,kick))

# Create a lookup dictionary
atoms = dict(zip([i[0] for i in out],range(len(out))))

# Create a coordinate dictionary
# This allows adding dummy particles for increasingly complex operations
coord = dict([(i[0],(_r(i[4],le-5),_r(i[5],le-5),_r(i[6],le-5))) for i in out])

# Correct terminal amino acid N/C positions, to increase stability of
# subsequent corrections.
if resn in _aminoacids and nterm:
    t = atoms.get("N")

    # Set N at 120 degree angle with respect to CA-C
    b = coord.get("CA")
    c = coord.get("C")

    # Only act if we really have backbone atoms
    if t != None and b != None and c != None:
        u = b[0]-c[0], b[1]-c[1], b[2]-c[2]
        l = math.sqrt(u[0]*u[0]+u[1]*u[1]+u[2]*u[2])
        try:
            v = _normalize(_crossprod(u,(u[0]+l,u[1],u[2])))
        except ZeroDivisionError:
            # Oh, so that vector was parallel. Then this must work.
            v = _normalize(_crossprod(u,(u[0],u[1]+l,u[2])))

        coord["N"] = (b[0]+u[0]/2+.866*1*v[0],b[1]+u[1]/2+.866*1*v[1],b[2]+u[2]/2+.866*1*v[2])
        out[t] = out[t][:4] + coord["N"]

if resn in _aminoacids and cterm:
    t = atoms.get("C")

    # Set N at 120 degree angle with respect to CA-N
    b = coord.get("CA")
    c = coord.get("N")

    # Only act if we really have backbone atoms
    if t != None and b != None and c != None:
        u = b[0]-c[0], b[1]-c[1], b[2]-c[2]
        l = math.sqrt(u[0]*u[0]+u[1]*u[1]+u[2]*u[2])
        try:
            v = _normalize(_crossprod(u,(u[0]+l,u[1],u[2])))
        except ZeroDivisionError:
            # Oh, so that vector was parallel. Then this must work.
            v = _normalize(_crossprod(u,(u[0],u[1]+l,u[2])))

        coord["C"] = (b[0]+u[0]/2+.866*1*v[0],b[1]+u[1]/2+.866*1*v[1],b[2]+u[2]/2+.866*1*v[2])
        out[t] = out[t][:4] + coord["C"]

# Before making modifications, save the raw projection results
raw = [i for i in out]

# Treat special cases: coordinate modifying operations
for tag,i in self.mod:

    if tag == "trans":

        #
        #      a          a--
        #     \         /  \
        #      b--c       b--c---d
        #           \         /
        #             d       e--(e+d+2c)
        #
        # a = b + c - d
        # Target particle
        a = i[0]

        # Get coordinates for the other atoms
        s = [ coord.get(j,-1) for j in i[1:] ]

        # Connecting particle
        b = s.pop(0)

        if min(s) > -1:
            # Subtract the third position from the fourth and so on, and sum the vectors
            u = _normalize((sum(k for k in zip(*[_normalize_vsub(j,s[0])] for j in s[1:] ))))

            # Get the new position
            x,y,z = b[0]-_normfac*u[0], b[1]-_normfac*u[1], b[2]-_normfac*u[2]

            # Index of target atom, if in the list
            t = atoms.get(a)

            # Set the coordinates in the output if this is a real particle
            if t != None:
                out[t] = out[t][:4] + (x,y,z)

            # Add coordinates to dictionary
            coord[a] = (x,y,z)
        else:
            print "Not all positions defined for [trans] operation:"
            print [(j,coord.get(j)) for j in i]

# The 'trans' definition is also a line of atom names
# Here the length should be four, always: a b c d
#try:
#    a,b,c,d = i
#except ValueError:
#    print "Invalid trans bond definition in residue %s (%s). Ignoring."%(out[0][1],i)
#    continue

# Source coordinates
#b = coord.get(b)
#c = coord.get(c)
#d = coord.get(d)

#if b != None and c != None and d != None:
#    # Vector c-d normalized
#    u = [_normfac*j for j in _normalize(vsub(c,d))]

```

```

# # New coordinates for a
# x,y,z = b[0]+u[0], b[1]+u[1], b[2]+u[2]

# # Index of target atom, if in the list
# t = atoms.get(a)

# # Set the coordinates in the output if this is a real particle
# if t != None:
#     out[t] = out[t][:4] + (x,y,z)

# # Add coordinates to dictionary
# coord[a] = (x,y,z)

elif tag == "cis":

#
#      b--c
#     /   \
#    a      d
#
# a = 2b - 2c + d

# The 'cis' definition is also a line of atom names
# Here the length should be four, always: a b c d
# The difference with trans dihedrals is in the equation below
try:
    a,b,c,d = i
except ValueError:
    print "Invalid trans bond definition in residue %(s). Ignoring."%(out[0][1],i)
    continue

# Source coordinates
b = coord.get(b)
c = coord.get(c)
d = coord.get(d)

if b != None and c != None and d != None:
    try:
        u = _normalize(_vsub(c,d))
        v = _normalize(_vsub(b,c))
        x,y,z = b[0]+_normfac*(v[0]-u[0]), b[1]+_normfac*(v[1]-u[1]), b[2]+_normfac*(v[2]-u[2])
    except ZeroDivisionError:
        x,y,z = 2*b[0]-2*c[0]+d[0], 2*b[1]-2*c[1]+d[1], 2*b[2]-2*c[2]+d[2]

    # Index of target atom, if in the list
    t = atoms.get(a)

    # Set the coordinates in the output if this is a real particle
    if t != None:
        out[t] = out[t][:4] + (x,y,z)

    # Add coordinates to dictionary
    coord[a] = (x,y,z)

elif tag == "out":

# Place a on the negative resultant vector
#
#      a
#     /
#    c--b
#     \
#      d
#
# a = 3b - c - d
#

# The 'out' definition is also a line of atom names
# But the length can be variable

# Target particle
a = i[0]

# Get coordinates for the other atoms
s = [ coord.get(j,-1) for j in i[1:] ]

if min(s) > -1:
    # Subtract the center from the other atoms and sum the vectors
    u = _normalize([sum(k) for k in zip(*[ _normalize(_vsub(j,s[0])) for j in s[1:] ])])

    # Get the new position
    x,y,z = s[0][0]-_normfac*u[0], s[0][1]-_normfac*u[1], s[0][2]-_normfac*u[2]

    # Index of target atom, if in the list
    t = atoms.get(a)

    # Set the coordinates in the output if this is a real particle
    if t != None:
        out[t] = out[t][:4] + (x,y,z)

    # Add coordinates to dictionary
    coord[a] = (x,y,z)

elif tag == "chiral":

# The 'chiral' definition is a list of atom names

# Target atom
a = i[0]

# Get coordinates for the other atoms; the first atom in this list is the chiral center
s = [ coord.get(j,-1) for j in i[1:] ]

if min(s) > -1:
    # Subtract the center from the other atoms
    u = [ _vsub(j,s[0]) for j in s[1:] ]

    # If there are multiple substituents given for a chiral center, determine the
    # average cross-product. Otherwise, use a more elaborate scheme for rebuilding.
    if len(u) > 2:

        # Get the resultant crossproduct normalized
        c = _normalize([sum(m) for m in zip(*[ _crossprod(j,k) for j,k in zip([u[-1]]+u,u) ])])

        # Set the new coordinates
        # Ai, magic number here! Where does the 0.1 come from?
        x,y,z = s[0][0]+_normfac*c[0], s[0][1]+_normfac*c[1], s[0][2]+_normfac*c[2]

    else:
#
#      a .
#     \ .
#    b--d
#     /
#      c
#
#      Like for CB/HA:

```



```

#           CB CA N C
#           HA CA C N
#           Or (safer):
#           CB CA N C
#           HA CA C N CB
#

# Unpack vectors
c, d = u

# Midpoint; a vector from the center b
p = [ .05*i for i in _normalize(_vadd(c,d)) ]

# Vector c-p
q = [ .05*i for i in _normalize(_vsub(c,d)) ]

# The vector in the direction of the target particle
try:
    w = [ _normfac*j for j in _normalize(_crossprod(q,p)) ]
except ZeroDivisionError:
    trm = nterm and ", N-terminus" or (cterm and ", C-terminus" or "")
    print "Chirality of %s (%s%s) for placing %s undefined by atoms %s. Skipping modification."%(i[1],resn,trm,i[0],repr(i[2:]))
    continue

# The coordinates
x,y,z = s[0][0]+w[0]-p[0], s[0][1]+w[1]-p[1], s[0][2]+w[2]-p[2]

# Index of target atom, if in the list
t = atoms.get(a)

# Set the coordinates in the output if this is a real particle
if t != None:
    out[t] = out[t][:4] + (x,y,z)

# Add coordinates to dictionary
coord[a] = (x,y,z)

# Now add a random kick again whenever needed to ensure that no atoms overlap
for i in range(len(out)):
    for j in range(i):
        if out[i][4] == out[j][4] and out[i][5] == out[j][5] and out[i][6] == out[j][6]:
            # Equal coordinates: need fix
            x,y,z = out[i][4:7]
            out[i] = out[i][:4]+(_r(x,kick),_r(y,kick),_r(z,kick))

return out, raw

# These are the modifier tags. They signify a specific
# operation on the atoms listed.
_mods = ("chiral","trans","cis","out")

# These are the default tags. Other tags should be
# coarse grained model names.
_tags = _mods + ("molecule","mapping","atoms")

def _init():
    molecules = []
    mapping = {}
    cg = []
    aa = []
    ff = []
    mod = []
    cur = []
    mol = []
    tag = re.compile('^ *[(.*) *]\')

# Read all .map residue definitions in the module directory
for filename in glob.glob(os.path.dirname(__file__)+"/*.map"):

    # Default CG model is martini.
    cg_ff = "martini"

    for line in open(filename):

        # Strip leading and trailing spaces
        s = line.strip()

        # Check for directive
        if s.startswith("["):

            # Extract the directive name
            cur = re.findall(tag,s)[0].strip().lower()

            if not cur in _tags: # cur == "martini":
                cg_ff = cur
                cg = []

            # The tag molecule starts a new molecule block
            # The tag mapping starts a new mapping block for a specific force field
            # In both cases, we need to purge the stuff that we have so far and
            # empty the variables, except 'mol'
            if cur in ("molecule","mapping"):
                # Check whether we have stuff
                # If so, we purge
                if aa:
                    for ffi in ff:
                        for m in mol:
                            try:
                                mapping[(m, ffi, cg_ff)] = ResidueMap(target=cg,atoms=aa,name=m)
                            except:
                                print "Error reading %s to %s mapping for %s (file: %s)."%(ffi,cg_ff,m,filename)
                            try:
                                mapping[(m, cg_ff, ffi)] = ResidueMap(atoms=aa,mod=mod,name=m)
                            except:
                                print "Error reading %s to %s mapping for %s (file: %s)."%(cg_ff,ffi,m,filename)

            # Reset lists
            aa,ff,mod = [],[],[]

            # Reset molecule name list if we have a molecule tag
            if cur == "molecule":
                mol = []

        continue

    # Remove comments
    s = s.split(';')[0].strip()

    if not s:
        # Skip empty lines
        continue

    elif cur == "molecule":
        mol.extend(s.split())

```

```

        elif not cur in _tags: # cur == "martini":
            # Martini coarse grained beads in topology order
            cg.extend(s.split())

        elif cur == "mapping":
            # Multiple force fields can be specified
            ff.extend(s.split())

        elif cur == "atoms":
            # Atom list for current force field molecule definition
            aa.append(s.split())

        elif cur in _mods:
            # Definitions of modifying operations
            mod.append((cur,s.split()))

# At the end we may still have rubbish left:
if aa:
    for ffi in ff:
        for m in mol:
            try:
                mapping[(m, ffi, cg_ff)] = ResidueMap(target=cg,atoms=aa,name=m)
            except:
                print "Error reading %s to %s mapping for %s (file: %s)."%(ffi,cg_ff,m,filename)
            try:
                mapping[(m, cg_ff, ffi)] = ResidueMap(atoms=aa,mod=mod,name=m)
            except:
                print "Error reading %s to %s mapping for %s (file: %s)."%(cg_ff,ffi,m,filename)

    return mapping

mapping = _init()

def get(target="gromos",source="martini"):
    D = dict([(i[0],mapping[i]) for i in mapping.keys() if i[1] == source and i[2] == target])
    print "Residues defined for transformation from %s to %s:"%(source,target)
    print D.keys()
    return D

__INIT_PY__

# ala.amber.map
cat << __MAP__ > ala.amber.map
[ molecule ]
ALA

[ martini ]
BB

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1 N BB
2 H BB
3 CA BB
4 HA BB
5 CB BB
6 HB1 BB
7 HB2 BB
8 HB3 BB
9 C BB
10 O BB

[ chiral ]
CB CA N C
HB1 CA N C
HB2 CA N C
HB3 CA N C

[ chiral ]
HA CA N CB C ; L-Ala
; HA CA N C CB ; D-Ala
__MAP__

# ala.charmm36.map
cat << __MAP__ > ala.charmm36.map
[ molecule ]
ALA

[ martini ]
BB

[ mapping ]
charmm27 charmm36

[ atoms ]
1 N BB
2 HN BB
3 CA BB
4 HA BB
5 CB BB
6 HB1 BB
7 HB2 BB
8 HB3 BB
9 C BB
10 O BB

[ chiral ]
CB CA N C
HB1 CA N C
HB2 CA N C
HB3 CA N C

[ chiral ]
HA CA N CB C ; L-Ala
; HA CA N C CB ; D-Ala
__MAP__

# ala.gromos.map
cat << __MAP__ > ala.gromos.map
[ molecule ]
ALA

[ martini ]
BB

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]

```

```

1      N      BB
2      H      BB
3      CA     BB
4      CB     BB
5      C      BB
6      O      BB

[ chiral ]
CB      CA      N      C
__MAP__

# aot.gromos.map
cat << __MAP__ > aot.gromos.map
[ molecule ]
AOT

[ martini ]
SO3 GL1 GL2 C1A C2A C1B C2B

[ mapping ]
gromos

[ atoms ]
; Charged head group (sulfonate)
1      CSC1  GL2
2      CES1  GL2
3      OES1  GL2
4      OSS1  GL2
5      C101  C1B
6      C102  C1B
7      C103  C2B
8      C104  C2B
9      C105  C2B
10     C106  C2B
11     C121  C1B
12     C122  C1B
13     CSC2  GL1
14     CES2  GL1
15     OES2  GL1
16     OSS2  GL1
17     C201  C1A
18     C202  C1A
19     C203  C2A
20     C204  C2A
21     C205  C2A
22     C206  C2A
23     C221  C1A
24     C222  C1A
25     SAO   SO3
26     OAO1  SO3
27     OAO2  SO3
28     OAO3  SO3
__MAP__

# arg.amber.map
cat << __MAP__ > arg.amber.map
[ molecule ]
ARG

[ martini ]
BB SC1 SC2

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1      N      BB
2      H      BB
3      CA     BB
4      HA     BB
5      CB      SC1  BB BB
6      HB1     SC1  BB BB
7      HB2     SC1  BB BB
8      CG      SC1  SC1 BB
9      HG1     SC1  SC1 BB
10     HG2     SC1  SC1 BB
11     CD      SC1
12     HD1     SC1
13     HD2     SC1
14     NE      SC2 SC1
15     HE      SC2 SC1
16     CZ      SC2
17     NH1     SC2
18     HH11    SC2
19     HH12    SC2
20     NH2     SC2
21     HH21    SC2
22     HH22    SC2
23     C       BB
24     O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Arg
; HA      CA      N      C      CB ; D-Arg

; The cis/trans are added to ensure proper
; splitting of the guanidinium group

[ trans ]
; Because of the use of normalized vectors, this makes sense:
NH1     CZ      NE      HE

[ out ]
NH2     CZ      NE      NH1
NH1     CZ      NE      NH2

[ out ]
HH11    NH1     CZ      HH12
HH12    NH1     CZ      HH11
HH21    NH2     CZ      HH22
HH22    NH2     CZ      HH21
__MAP__

# arg.charmm36.map
cat << __MAP__ > arg.charmm36.map
[ molecule ]
ARG

[ martini ]
BB SC1 SC2

```

```

[ mapping ]
charmm27 charmm36

[ atoms ]
  1      N      BB
  2      HN     BB
  3      CA     BB
  4      HA     BB
  5      CB     SC1 BB BB
  6      HB1    SC1 BB BB
  7      HB2    SC1 BB BB
  8      CG     SC1 SC1 BB
  9      HG1    SC1 SC1 BB
 10     HG2    SC1 SC1 BB
 11      CD     SC1
 12     HD1    SC1
 13     HD2    SC1
 14      NE     SC2 SC1
 15      HE     SC2 SC1
 16      CZ     SC2
 17     NH1    SC2
 18     HH11   SC2
 19     HH12   SC2
 20     NH2    SC2
 21     HH21   SC2
 22     HH22   SC2
 23      C      BB
 24      O      BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Arg
; HA     CA      N      C      CB ; D-Arg

; The cis/trans are added to ensure proper
; splitting of the guanidinium group

[ trans ]
; Because of the use of normalized vectors, this makes sense:
NH1     CZ      NE      HE

[ out ]
NH2     CZ      NE      NH1
NH1     CZ      NE      NH2

[ out ]
HH11    NH1     CZ      HH12
HH12    NH1     CZ      HH11
HH21    NH2     CZ      HH22
HH22    NH2     CZ      HH21
__MAP__

# arg.gromos.map
cat << __MAP__ > arg.gromos.map
[ molecule ]
ARG

[ martini ]
BB SC1 SC2

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA     BB
  4      CB     SC1 BB BB
  5      CG     SC1 SC1 BB
  6      CD     SC1
  7      NE     SC2 SC1
  8      HE     SC2 SC1
  9      CZ     SC2
 10     NH1    SC2
 11     HH11   SC2
 12     HH12   SC2
 13     NH2    SC2
 14     HH21   SC2
 15     HH22   SC2
 16      C      BB
 17      O      BB

; The cis/trans are added to ensure proper
; splitting of the guanidinium group

[ trans ]
; Because of the use of normalized vectors, this makes sense:
; CD      CG      NE      HE
; HE      NE      CZ      NH1
; NH1     CZ      NE      HE

[ out ]
NH2     CZ      NE      NH1
NH1     CZ      NE      NH2

; ---
; HH11    NH1     CZ      NE
; HH12    NH1     CZ      NH2
; HH21    NH2     CZ      NE
; HH22    NH2     CZ      NH1

[ out ]
HH11    NH1     CZ      HH12
HH12    NH1     CZ      HH11
HH21    NH2     CZ      HH22
HH22    NH2     CZ      HH21

[ chiral ]
CB      CA      N      C

;[ cis ]
; NH2     CZ      NE      CD
; HH12    NH1     CZ      NE
; HH21    NH2     CZ      NE
__MAP__

# asn.amber.map
cat << __MAP__ > asn.amber.map
[ molecule ]
ASN

[ martini ]

```

```

BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      CG      SC1 SC1 BB
  9      OD1     SC1
 10      ND2     SC1
 11      HD21    SC1
 12      HD22    SC1
 13      C       BB
 14      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Asn
; HA     CA      N      C      CB ; D-Asn
__MAP__

# asn.charmm36.map
cat << __MAP__ > asn.charmm36.map
[ molecule ]
ASN

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
  1      N      BB
  2      HN      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      CG      SC1 SC1 BB
  9      OD1     SC1
 10      ND2     SC1
 11      HD21    SC1
 12      HD22    SC1
 13      C       BB
 14      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Asn
; HA     CA      N      C      CB ; D-Asn
__MAP__

# asn.gromos.map
cat << __MAP__ > asn.gromos.map
[ molecule ]
ASN

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      CB      SC1 BB BB
  5      CG      SC1 SC1 BB
  6      OD1     SC1
  7      ND2     SC1
  8      HD21    SC1
  9      HD22    SC1
 10      C       BB
 11      O       BB

[ chiral ]
CB      CA      N      C
__MAP__

# asp.amber.map
cat << __MAP__ > asp.amber.map
[ molecule ]
ASP

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      CG      SC1 SC1 BB
  9      OD1     SC1
 10      OD2     SC1
 11      HD2     SC1
 12      C       BB
 13      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

```

```

[ chiral ]
HA    CA    N    CB    C ; L-Asp
; HA    CA    N    C    CB ; D-Asp
__MAP__

# asp.charmm36.map
cat << __MAP__ > asp.charmm36.map
[ molecule ]
ASP

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN     BB
3      CA     BB
4      HA     BB
5      CB     SC1 BB BB
6      HB1    SC1 BB BB
7      HB2    SC1 BB BB
8      CG     SC1 SC1 BB
9      OD1    SC1
10     OD2    SC1
11     HD2    SC1
12     C      BB
13     O      BB

[ chiral ]
CB     CA     N     C
HB1    CA     N     C
HB2    CA     N     C

[ chiral ]
HA     CA     N     CB    C ; L-Asp
; HA     CA     N     C    CB ; D-Asp
__MAP__

# asp.gromos.map
cat << __MAP__ > asp.gromos.map
[ molecule ]
ASP

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA     BB
4      CB     SC1 BB BB
5      CG     SC1 SC1 BB
6      OD1    SC1
7      OD2    SC1
7      HD2    SC1
8      C      BB
9      O      BB

[ chiral ]
CB     CA     N     C
__MAP__

# chol.amber.map
cat << __MAP__ > chol.amber.map
; Slipids by Joakim P. M. Jämbäck made for combining with Amber force field
; in Slipids cholesterol is called CHLI
[ molecule ]
CHOL

[ martini ]
ROH R1 R2 R3 R4 R5 C1 C2

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1      C3     ROH ROH ROH R1
2      O3     ROH
3      H3'    ROH
4      H3     ROH
5      C4     ROH ROH ROH R2
6      H4A    ROH ROH ROH R2
7      H4B    ROH ROH ROH R2
8      C5     R1  ROH ROH R2 R2
9      C6     R2 R2 R2 R2 ROH
10     H6     R2 R2 R2 R2 ROH
11     C7     R2
12     H7A    R2
13     H7B    R2
14     C8     R2 R2 R3 R4
15     H8     R2 R2 R3 R4
16     C14    R4 R4 R3
17     H14    R4
18     C15    R4
19     H15A   R4
20     H15B   R4
21     C16    R4 R4 C1
22     H16A   R4 R4 C1
23     H16B   R4 R4 C1
24     C17    R5 R4 C1 C1
25     H17    R5 R3 R4 C1 C1
26     C13    R5
27     C18    R5
28     H18A   R5
29     H18B   R5
30     H18C   R5
31     C12    R3 R3 R3 C1
32     H12A   R3 R3 R3 C1
33     H12B   R3 R3 R3 C1
34     C11    R3
35     H11A   R3
36     H11B   R3
37     C9     R3 R3 R3 R2 ROH
38     H9     R3 R3 R3 R2 ROH
39     C10    R1 R1
40     C19    R1
41     H19A   R1
42     H19B   R1
43     H19C   R1

```

```

44 C1 R1
45 H1A R1
46 H1B R1
47 C2 ROH
48 H2A ROH
49 H2B ROH
50 C20 C1 C1 C1 R4
51 H20 C1
52 C21 C1
53 H21A C1
54 H21B C1
55 H21C C1
56 C22 C1
57 H22A C1
58 H22B C1
59 C23 C1 C1 C1 C2
60 H23A C1 C1 C1 C2
61 H23B C1 C1 C1 C2
62 C24 C2 C1
63 H24A C2 C1
64 H24B C2 C1
65 C25 C2 C2 C2 C1
66 H25 C2 C2 C2 C1
67 C26 C2
68 H26A C2
69 H26B C2
70 H26C C2
71 C27 C2
72 H27A C2
73 H27B C2
74 H27C C2

[chiral]
H3 C3 C4 O3 C2
H8 C8 C7 C14 C9
H14 C14 C15 C8 C13
H9 C9 C10 C11 C8
H17 C17 C16 C13 C20

[out]
C21 C20 C22 C17
H21A C20 C22 C17
H21B C20 C22 C17
H21C C20 C22 C17
[chiral]
H20 C20 C17 C21 C22

[chiral]
C18 C13 C12 C14 C17
H18A C13 C12 C14 C17
H18B C13 C12 C14 C17
H18C C13 C12 C14 C17

[out]
C2 C3 O3 C4
H2A C3 O3 C4
H2B C3 O3 C4
[trans]
C1 C2 C3 O3
H1A C2 C3 O3
H1B C2 C3 O3

[chiral]
C19 C10 C1 C5 C9
H19A C10 C1 C5 C9
H19B C10 C1 C5 C9
H19C C10 C1 C5 C9
__MAP__

# chol.charmm36.map
cat << __MAP__ > chol.charmm36.map
; latest revision Kri 12.3.2013
[ molecule ]
CHOL

[ martini ]
ROH R1 R2 R3 R4 R5 C1 C2

[ mapping ]
charmm27 charmm36

[ atoms ]
1 C1 ROH ROH ROH R1
2 O2 ROH
3 H3 ROH
4 H4 ROH
5 C5 ROH ROH ROH R2
6 H6 ROH ROH ROH R2
7 H7 ROH ROH ROH R2
8 C8 R1 ROH ROH R2 R2
9 C9 R2 R2 R2 R2 ROH
10 H10 R2 R2 R2 R2 ROH
11 C11 R2
12 H12 R2
13 H13 R2
14 C14 R2 R2 R3 R4
15 H15 R2 R2 R3 R4
16 C16 R4 R4 R3
17 H17 R4
18 C18 R4
19 H19 R4
20 H20 R4
21 C21 R4 R4 C1
22 H22 R4 R4 C1
23 H23 R4 R4 C1
24 C24 R5 R4 C1 C1
25 H25 R5 R3 R4 C1 C1
26 C26 R5
27 C27 R5
28 H28 R5
29 H29 R5
30 H30 R5
31 C31 R3 R3 R3 C1
32 H32 R3 R3 R3 C1
33 H33 R3 R3 R3 C1
34 C34 R3
35 H35 R3
36 H36 R3
37 C37 R3 R3 R3 R2 ROH
38 H38 R3 R3 R3 R2 ROH
39 C39 R1 R1
40 C40 R1
41 H41 R1
42 H42 R1
43 H43 R1
44 C44 R1
45 H45 R1
46 H46 R1

```

```

47 C47 ROH
48 H48 ROH
49 H49 ROH
50 C50 C1 C1 C1 R4
51 H51 C1
52 C52 C1
53 H53 C1
54 H54 C1
55 H55 C1
56 C56 C1
57 H57 C1
58 H58 C1
59 C59 C1 C1 C1 C2
60 H60 C1 C1 C1 C2
61 H61 C1 C1 C1 C2
62 C62 C2 C1
63 H63 C2 C1
64 H64 C2 C1
65 C65 C2 C2 C2 C1
66 H66 C2 C2 C2 C1
67 C67 C2
68 H68 C2
69 H69 C2
70 H70 C2
71 C71 C2
72 H72 C2
73 H73 C2
74 H74 C2

[chiral]
H4 C1 C5 O2 C47
H15 C14 C11 C16 C37
H17 C16 C18 C14 C26
H38 C37 C39 C34 C14
H25 C24 C21 C26 C50
[out]
C52 C50 C56 C24
H53 C50 C56 C24
H54 C50 C56 C24
H44 C50 C56 C24
[chiral]
H51 C50 C24 C52 C56

[chiral]
C27 C26 C31 C16 C24
H28 C26 C31 C16 C24
H29 C26 C31 C16 C24
H30 C26 C31 C16 C24

[out]
C47 C1 O2 C5
H48 C1 O2 C5
H49 C1 O2 C5
[trans]
C44 C47 C1 O2
H45 C47 C1 O2
H46 C47 C1 O2

[chiral]
C40 C39 C44 C8 C37
H41 C39 C44 C8 C37
H42 C39 C44 C8 C37
H43 C39 C44 C8 C37
__MAP__

# chol.gromos.map
cat << __MAP__ > chol.gromos.map
[ molecule ]
CHOL

[ martini ]
ROH R1 R2 R3 R4 R5 C1 C2

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1 C1 R1
2 C2 R1
3 C3 R1
4 C4 ROH
5 C5 ROH ROH ROH R1
6 O6 ROH
7 H ROH
8 C8 ROH ROH ROH R2
9 C9 R1 ROH ROH R2 R2
10 C10 R2 R2 R2 R2 ROH
11 C11 R2
12 C12 R2 R2 R3 R4
13 C13 R3 R3 R3 ROH R2
14 C14 R3
15 C15 R3 R3 R3 C1
16 C16 R5
17 C17 R5
18 C18 R4 R4 R3
19 C19 R4
20 C20 R4 R4 C1
21 C21 R5 R3 R4 C1 C1
22 C22 C1 C1 C1 R4
23 C23 C1 C1 C1 R3
24 C24 C1
25 C25 C1 C1 C1 C2
26 C26 C2 C1
27 C27 C2 C2 C2 C1
28 C28 C2
29 C29 C2

[out]
C4 C5 O6 C8
[trans]
C3 C4 C5 O6

[chiral]
C1 C2 C3 C9 C13
[ chiral ]
C17 C16 C18 C21 C15
__MAP__

# c12h.gromos.map
cat << __MAP__ > c12h.gromos.map
[ molecule ]
CL2H

[ martini ]
GL5 FO41 GL1 GL2 C1A C2A D3A C4A C5A C1B C2B D3B C4B C5B FO42 GL3 GL4 C1A2 C2A2 D3A2 C4A2 C5A2 C1B2 C2B2 D3B2 C4B2 C5B2

```



```

[ gromos53a6 ]
  1 CGS1  GL4
  2 OGS1  GL4
  3 CES1  GL4
  4 OES1  GL4
  5 C102  C1B2
  6 C103  C1B2
  7 C104  C1B2
  8 C105  C1B2
  9 C106  C2B2
 10 C107  C2B2
 11 C108  C2B2
 12 C109  D3B2
 13 C110  D3B2
 14 C111  D3B2
 15 C112  C4B2
 16 C113  C4B2
 17 C114  C4B2
 18 C115  C5B2
 19 C116  C5B2
 20 C117  C5B2
 21 C118  C5B2
 22 CGS2  GL3
 23 OGS2  GL3
 24 CES2  GL3
 25 OES2  GL3
 26 C202  C1A2
 27 C203  C1A2
 28 C204  C1A2
 29 C205  C1A2
 30 C206  C2A2
 31 C207  C2A2
 32 C208  C2A2
 33 C209  D3A2
 34 C210  D3A2
 35 C211  D3A2
 36 C212  C4A2
 37 C213  C4A2
 38 C214  C4A2
 39 C215  C5A2
 40 C216  C5A2
 41 C217  C5A2
 42 C218  C5A2
 43 CGS3  GL3
 44 OGS3  P041
 45 PCL1  P041
 46 OCL1  P041
 47 OCL2  P041
 48 OCL3  P041
 49 CCL1  GL5
 50 CCL2  GL5
 51 OCL4  GL5
 52 HCL4  GL5
 53 CCL3  GL5
 54 OCL5  P042
 55 PCL2  P042
 56 OCL6  P042
 57 OCL7  P042
 58 OCL8  P042
 59 CGS1  GL4
 60 OGS1  GL4
 61 CES1  GL4
 62 OES1  GL4
 63 C102  C1B2
 64 C103  C1B2
 65 C104  C1B2
 66 C105  C1B2
 67 C106  C2B2
 68 C107  C2B2
 69 C108  C2B2
 70 C109  D3B2
 71 C110  D3B2
 72 C111  D3B2
 73 C112  C4B2
 74 C113  C4B2
 75 C114  C4B2
 76 C115  C5B2
 77 C116  C5B2
 78 C117  C5B2
 79 C118  C5B2
 80 CGS2  GL3
 81 OGS2  GL3
 82 CES2  GL3
 83 OES2  GL3
 84 C202  C1A2
 85 C203  C1A2
 86 C204  C1A2
 87 C205  C1A2
 88 C206  C2A2
 89 C207  C2A2
 90 C208  C2A2
 91 C209  D3A2
 92 C210  D3A2
 93 C211  D3A2
 94 C212  C4A2
 95 C213  C4A2
 96 C214  C4A2
 97 C215  C5A2
 98 C216  C5A2
 99 C217  C5A2
100 C218  C5A2
101 CGS3  GL3
__MAP__

# c14.gromos.map
cat << __MAP__ > c14.gromos.map
[ molecule ]
CL4

[ martini ]
GL5 P041 GL1 GL2 C1A C2A D3A C4A C5A C1B C2B D3B C4B C5B P042 GL3 GL4 C1A2 C2A2 D3A2 C4A2 C5A2 C1B2 C2B2 D3B2 C4B2 C5B2

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1 CGS1  GL2 GL2 GL1
  2 OGS1  GL2 GL2 GL1 C1B
  3 CES1  GL2 GL2 C1B
  4 OES1  GL2 GL2 GL2 C1B
  5 C102  C1B C1B GL2
  6 C103  C1B
  7 C104  C1B C1B C2B
  8 C105  C1B C2B
  9 C106  C2B C2B C1B
 10 C107  C2B
 11 C108  C2B D3B
 12 C109  D3B D3B C2B

```

```

13 C110 D3B D3B C4B
14 C111 D3B C4B
15 C112 C4B
16 C113 C4B C4B C4B C5B
17 C114 C4B C4B C5B
18 C115 C5B C4B
19 C116 C5B C5B C4B
20 C117 C5B C5B C5B C4B
21 C118 C5B
22 CGS2 GL1 GL1 GL2 C1A
23 OGS2 GL1 GL1 GL2
24 CES2 GL1 GL1 C1A
25 OES2 GL1 GL1 GL1 C1A
26 C202 C1A C1A GL1
27 C203 C1A
28 C204 C1A C1A C2A
29 C205 C1A C2A
30 C206 C2A C2A C1A
31 C207 C2A
32 C208 C2A D3A
33 C209 D3A D3A C2A
34 C210 D3A D3A C4A
35 C211 D3A C4A
36 C212 C4A
37 C213 C4A C4A C4A C5A
38 C214 C4A C4A C5A
39 C215 C5A C4A
40 C216 C5A C5A C4A
41 C217 C5A C5A C5A C4A
42 C218 C5A
43 CGS3 GL1 GL1 P041
44 OGS3 P041 P041 GL1
45 PCL1 P041
46 OCL1 P041
47 OCL2 P041
48 OCL3 P041 P041 GL5
49 CCL1 GL5 GL5 P041
50 CCL2 GL5 P041 P042
51 OCL4 GL5
52 HCL4 GL5
53 CCL3 GL5 GL5 P042
54 OCL5 P042 P042 GL5
55 PCL2 P042
56 OCL6 P042
57 OCL7 P042
58 OCL8 P042 P042 GL3
59 CGS4 GL4 GL4 GL3
60 OGS4 GL4 GL4 GL3 C1B2
61 CES4 GL4 GL4 C1B2
62 OES4 GL4 GL4 GL4 C1B2
63 C302 C1B2 C1B2 GL4 GL4 GL4
64 C303 C1B2
65 C304 C1B2 C1B2 C2B2
66 C305 C1B2 C2B2
67 C306 C2B2 C2B2 C1B2
68 C307 C2B2
69 C308 C2B2 D3B2
70 C309 D3B2 D3B2 C2B2
71 C310 D3B2 D3B2 C4B2
72 C311 D3B2 C4B2
73 C312 C4B2
74 C313 C4B2 C4B2 C4B2 C5B2
75 C314 C4B2 C4B2 C5B2
76 C315 C5B2 C4B2
77 C316 C5B2 C5B2 C4B2
78 C317 C5B2 C5B2 C5B2 C4B2
79 C318 C5B2
80 CGS5 GL3 GL3 GL4
81 OGS5 GL3 GL3 GL4 C1A2
82 CES5 GL3 GL3 C1A2
83 OES5 GL3 GL3 GL3 C1A2
84 C402 C1A2 C1A2 GL3
85 C403 C1A2
86 C404 C1A2 C1A2 C2A2
87 C405 C1A2 C2A2
88 C406 C2A2 C2A2 C1A2
89 C407 C2A2
90 C408 C2A2 D3A2
91 C409 D3A2 D3A2 C2A2
92 C410 D3A2 D3A2 C4A2
93 C411 D3A2 C4A2
94 C412 C4A2
95 C413 C4A2 C4A2 C4A2 C5A2
96 C414 C4A2 C4A2 C5A2
97 C415 C5A2 C4A2
98 C416 C5A2 C5A2 C4A2
99 C417 C5A2 C5A2 C5A2 C4A2
100 C418 C5A2
101 CGS6 GL3 GL3 P042
__MAP__

```

```

# cys.amber.map
cat << __MAP__ > cys.amber.map
[ molecule ]
CYS

```

```

[ martini ]
BB SC1

```

```

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

```

```

[ atoms ]
1 N BB
2 H BB
3 CA BB
4 HA BB
5 CB SC1 BB
6 HB1 SC1 BB
7 SG SC1
8 HG SC1
9 C BB
10 O BB

```

```

[ chiral ]
CB CA N C
HB1 CA N C
HB2 CA N C

```

```

[ chiral ]
HA CA N CB C ; L-Cys
; HA CA N C CB ; D-Cys
__MAP__

```

```

# cys.charmm36.map
cat << __MAP__ > cys.charmm36.map

```

```

[ molecule ]
CYS

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN     BB
3      CA     BB
3      HA     BB
4      CB     SC1 BB
5      HB1    SC1 BB
6      HB2    SC1 BB
7      SG     SC1
8      HG     SC1
9      C      BB
10     O      BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Cys
; HA    CA      N      C      CB ; D-Cys
__MAP__

# cys.gromos.map
cat << __MAP__ > cys.gromos.map
[ molecule ]
CYS

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA     BB
4      CB     SC1 BB
5      SG     SC1
6      HG     SC1
7      C      BB
8      O      BB

[ chiral ]
CB      CA      N      C
__MAP__

# dopc.amber.map
cat << __MAP__ > dopc.amber.map
; Slipids by Joakim P. M. Jämbeck made for combining with amber force fields

[ molecule ]
DOPC
;
; NC3-PO4-GL1-C1A-C2A-D3A-C4A-C5A
; |
; GL2-C1B-C2B-D3B-C4B-C5B

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A D3A C4A C5A C1B C2B D3B C4B C5B

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
; Terminal head group (choline)
1      N      NC3
2      C12    NC3 NC3 NC3 PO4
3      C13    NC3
4      C14    NC3
5      C15    NC3
6      H12A   NC3 NC3 NC3 PO4
7      H12B   NC3 NC3 NC3 PO4
8      H13A   NC3
9      H13B   NC3
10     H13C   NC3
11     H14A   NC3
12     H14B   NC3
13     H14C   NC3
14     H15A   NC3
15     H15B   NC3
16     H15C   NC3
17     C11    NC3 PO4
18     H11A   NC3 PO4
19     H11B   NC3 PO4
; Phosphate group
20     P      PO4
21     O13    PO4
22     O14    PO4
23     O12    PO4 PO4 PO4 NC3
24     O11    PO4 PO4 GL1
; Diacylglycerol
25     C1     GL1 GL1 PO4
26     HA     GL1 GL1 PO4
27     HB     GL1 GL1 PO4
28     C2     GL1 GL1 GL2
29     HS     GL1 GL1 GL2
30     O21    GL1 GL1 GL2 C1A
31     C21    GL1 C1A
32     O22    GL1
33     C22    C1A C1A GL1
34     H2R    C1A C1A GL1
35     H2S    C1A C1A GL1
36     C3     GL2 GL2 GL2 PO4
37     HX     GL2 GL2 GL2 PO4
38     HY     GL2 GL2 GL2 PO4
39     O31    GL2
40     C31    GL2 GL2 C1B
41     O32    GL2
42     C32    C1B C1B GL2
43     H2X    C1B C1B GL2
44     H2Y    C1B C1B GL2
45     C23    C1A
46     H3R    C1A
47     H3S    C1A
48     C24    C1A C1A C2A
49     H4R    C1A C1A C2A

```

```

50 H4S C1A C1A C2A
51 C25 C1A C2A
52 H5R C1A C2A
53 H5S C1A C2A
54 C26 C2A C2A C1A
55 H6R C2A C2A C1A
56 H6S C2A C2A C1A
57 C27 C2A
58 H7R C2A
59 H7S C2A
60 C28 C2A D3A
61 H8R C2A D3A
62 H8S C2A D3A
63 C29 D3A D3A C2A
64 H9R D3A
65 C210 D3A D3A C4A
66 H10R D3A
67 C211 D3A
68 H11R D3A C4A
69 H11S D3A C4A
70 C212 C4A
71 H12R C4A
72 H12S C4A
73 C213 C4A C4A C4A C5A
74 H13R C4A C4A C4A C5A
75 H13S C4A C4A C4A C5A
76 C214 C4A C4A C5A
77 H14R C4A C4A C5A
78 H14S C4A C4A C5A
79 C215 C5A C4A
80 H15R C5A C4A
81 H15S C5A C4A
82 C216 C5A C5A C4A
83 H16R C5A C5A C4A
84 H16S C5A C5A C4A
85 C217 C5A C5A C5A C4A
86 H17R C5A C5A C5A C4A
87 H17S C5A C5A C5A C4A
88 C218 C5A
89 H18R C5A
90 H18S C5A
91 H18T C5A
92 C33 C1B
93 H3X C1B
94 H3Y C1B
95 C34 C1B C1B C2B
96 H4X C1B C1B C2B
97 H4Y C1B C1B C2B
98 C35 C1B C2B
99 H5X C1B C2B
100 H5Y C1B C2B
101 C36 C2B C2B C1B
102 H6X C2B C2B C1B
103 H6Y C2B C2B C1B
104 C37 C2B
105 H7X C2B
106 H7Y C2B
107 C38 C2B D3B
108 H8X C2B D3B
109 H8Y C2B D3B
110 C39 D3B D3B C2B
111 H9X D3B D3B C2B
112 C310 D3B D3B C4B
113 H10X D3B D3B C4B
114 C311 D3B C4B
115 H11X D3B C4B
116 H11Y D3B C4B
117 C312 C4B
118 H12X C4B
119 H12Y C4B
120 C313 C4B C4B C4B C5B
121 H13X C4B C4B C4B C5B
122 H13Y C4B C4B C4B C5B
123 C314 C4B C4B C5B
124 H14X C4B C4B C5B
125 H14Y C4B C4B C5B
126 C315 C5B C4B
127 H15X C5B C4B
128 H15Y C5B C4B
129 C316 C5B C5B C4B
130 H16X C5B C5B C4B
131 H16Y C5B C5B C4B
132 C317 C5B C5B C5B C4B
133 H17X C5B C5B C5B C4B
134 H17Y C5B C5B C5B C4B
135 C318 C5B
136 H18X C5B
137 H18Y C5B
138 H18Z C5B

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipid chains
[ out ]
H9R C29 C28 C210
[ trans ]
H10R C210 C29 C28
[ out ]
C211 C210 C29 H10R

[ out ]
H9X C39 C38 C310
[ trans ]
H10X C310 C39 C38
[ out ]
C311 C310 C39 H10X

;;;making a choline group
[out]
C14 N C13 C12
H14A N C13 C12
H14B N C13 C12
H14C N C13 C12

[ chiral ]
C15 N C12 C13 C14
H15A N C12 C13 C14

```

```

H15B N C12 C13 C14
H15C N C12 C13 C14
__MAP__

# dopc.charmm36.map
cat << __MAP__ > dopc.charmm36.map
; Mapping file created 15/03/2013 by Kri

[ molecule ]
DOPC
;
; NC3-PO4-GL1-C1A-C2A-D3A-C4A-C5A
;
; GL2-C1B-C2B-D3B-C4B-C5B

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A D3A C4A C5A C1B C2B D3B C4B C5B

[ mapping ]
charmm27 charmm36

[ atoms ]
; Terminal head group (choline)
1 N NC3
2 C12 NC3 NC3 NC3 PO4
3 C13 NC3
4 C14 NC3
5 C15 NC3
6 H12A NC3 NC3 NC3 PO4
7 H12B NC3 NC3 NC3 PO4
8 H13A NC3
9 H13B NC3
10 H13C NC3
11 H14A NC3
12 H14B NC3
13 H14C NC3
14 H15A NC3
15 H15B NC3
16 H15C NC3
17 C11 NC3 PO4
18 H11A NC3 PO4
19 H11B NC3 PO4
; Phosphate group
20 P PO4
21 O13 PO4
22 O14 PO4
23 O12 PO4 PO4 PO4 NC3
24 O11 PO4 PO4 GL1
; Diacylglycerol
25 C1 GL1 GL1 PO4
26 HA GL1 GL1 PO4
27 HB GL1 GL1 PO4
28 C2 GL1 GL1 GL2
29 HS GL1 GL1 GL2
30 O21 GL1 GL1 GL2 C1A
31 C21 GL1 C1A
32 O22 GL1
33 C22 C1A C1A GL1
34 H2R C1A C1A GL1
35 H2S C1A C1A GL1
36 C3 GL2 GL2 GL2 PO4
37 HX GL2 GL2 GL2 PO4
38 HY GL2 GL2 GL2 PO4
39 O31 GL2
40 C31 GL2 GL2 C1B
41 O32 GL2
42 C32 C1B C1B GL2
43 H2X C1B C1B GL2
44 H2Y C1B C1B GL2
45 C23 C1A
46 H3R C1A
47 H3S C1A
48 C24 C1A C1A C2A
49 H4R C1A C1A C2A
50 H4S C1A C1A C2A
51 C25 C1A C2A
52 H5R C1A C2A
53 H5S C1A C2A
54 C26 C2A C2A C1A
55 H6R C2A C2A C1A
56 H6S C2A C2A C1A
57 C27 C2A
58 H7R C2A
59 H7S C2A
60 C28 C2A D3A
61 H8R C2A D3A
62 H8S C2A D3A
63 C29 D3A D3A C2A
64 H9R D3A
65 C210 D3A D3A C4A
66 H10R D3A
67 C211 D3A
68 H11R D3A C4A
69 H11S D3A C4A
70 C212 C4A
71 H12R C4A
72 H12S C4A
73 C213 C4A C4A C4A C5A
74 H13R C4A C4A C4A C5A
75 H13S C4A C4A C4A C5A
76 C214 C4A C4A C5A
77 H14R C4A C4A C5A
78 H14S C4A C4A C5A
79 C215 C5A C4A
80 H15R C5A C4A
81 H15S C5A C4A
82 C216 C5A C5A C4A
83 H16R C5A C5A C4A
84 H16S C5A C5A C4A
85 C217 C5A C5A C5A C4A
86 H17R C5A C5A C5A C4A
87 H17S C5A C5A C5A C4A
88 C218 C5A
89 H18R C5A
90 H18S C5A
91 H18T C5A
92 C33 C1B
93 H3X C1B
94 H3Y C1B
95 C34 C1B C1B C2B
96 H4X C1B C1B C2B
97 H4Y C1B C1B C2B
98 C35 C1B C2B
99 H5X C1B C2B
100 H5Y C1B C2B
101 C36 C2B C2B C1B
102 H6X C2B C2B C1B
103 H6Y C2B C2B C1B

```

```

104 C37 C2B
105 H7X C2B
106 H7Y C2B
107 C38 C2B D3B
108 H8X C2B D3B
109 H8Y C2B D3B
110 C39 D3B D3B C2B
111 H9X D3B D3B C2B
112 C310 D3B D3B C4B
113 H10X D3B D3B C4B
114 C311 D3B C4B
115 H11X D3B C4B
116 H11Y D3B C4B
117 C312 C4B
118 H12X C4B
119 H12Y C4B
120 C313 C4B C4B C4B C5B
121 H13X C4B C4B C4B C5B
122 H13Y C4B C4B C4B C5B
123 C314 C4B C4B C5B
124 H14X C4B C4B C5B
125 H14Y C4B C4B C5B
126 C315 C5B C4B
127 H15X C5B C4B
128 H15Y C5B C4B
129 C316 C5B C5B C4B
130 H16X C5B C5B C4B
131 H16Y C5B C5B C4B
132 C317 C5B C5B C5B C4B
133 H17X C5B C5B C5B C4B
134 H17Y C5B C5B C5B C4B
135 C318 C5B
136 H18X C5B
137 H18Y C5B
138 H18Z C5B

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipind chains
[ out ]
H9R C29 C28 C210
[ trans ]
H10R C210 C29 C28
[ out ]
C211 C210 C29 H10R

[ out ]
H9X C39 C38 C310
[ trans ]
H10X C310 C39 C38
[ out ]
C311 C310 C39 H10X

;;;making a choline group
[out]
C14 N C13 C12
H14A N C13 C12
H14B N C13 C12
H14C N C13 C12

[ chiral ]
C15 N C12 C13 C14
H15A N C12 C13 C14
H15B N C12 C13 C14
H15C N C12 C13 C14
__MAP__

# dopc.gromos.map
cat << __MAP__ > dopc.gromos.map
; Created by Kri on 6.3.2013
[ molecule ]
DOPC

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A D3A C4A C5A C1B C2B D3B C4B C5B

;
; NC3-PO4-GL1-C1A-C2A-D3A-C4A-C5A
; |
; GL2-C1B-C2B-D3B-C4B-C5A

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
; Terminal head group (choline)
1 CN1 NC3
2 CN2
3 CN3
4 NTM
5 CA NC3 NC3 PO4
6 CB NC3 PO4
; Phosphate group
7 OA PO4 PO4 NC3
8 P PO4
9 OB
10 OC
11 OD PO4 PO4 GL1
; Diacylglycerol
12 CC GL1 GL1 PO4
13 CD GL1
14 OE GL1 GL1 GL1 C1A
15 C1A GL1 C1A
16 OF
17 C1B C1A C1A C1A GL1
18 C1C C1A
19 C1D C1A C1A C2A
20 C1E C1A C2A
21 C1F C2A C2A C1A
22 C1G C2A
23 C1H C2A D3A
24 C1I D3A D3A C2A
25 C1J D3A D3A C4A
26 C1K D3A C4A
27 C1L C4A

```

```

28 C1M C4A C4A C4A C5A
29 C1N C4A C4A C4A C5A
30 C1O C5A C4A
31 C1P C5A C5A C4A
32 C1Q C5A C5A C5A C4A
33 C1R C5A
34 CE GL2
35 OG GL2 GL2 GL2 C1B
36 C2A GL2 C1B
37 OH
38 C2B C1B C1B C1B GL2
39 C2C C1B
40 C2D C1B C1B C1B C2B
41 C2E C1B C2B
42 C2F C2B C2B C2B C1B
43 C2G C2B
44 C2H C2B D3B
45 C2I D3B D3B C2B
46 C2J D3B D3B C4B
47 C2K D3B C4B
48 C2L C4B
49 C2M C4B C4B C4B C5B
50 C2N C4B C4B C5B
51 C2O C5B C4B
52 C2P C5B C5B C4B
53 C2Q C5B C5B C4B
54 C2R C5B

; Choline group
[out]
;CN2 NTM CN1 CA
[chiral]
;CN3 NTM CN1 CN2 CA

; Cis double bonds
[ cis ]
C1H C1I C1J C1K
C2H C2I C2J C2K

; Acyl esters
; =====
; This reconstruction is somewhat complex. Unfortunately
; the Gromos united atom force field does not have
; correct dihedrals for acyl esters and these groups
; have to be built with correct geometry. Only setting
; the C-O-CO-C dihedrals correct is not sufficient, as
; the distortions may be so large that the dihedrals
; are pushed over the barrier. Therefore, the whole
; glycerol group is rebuilt so as to form a buffer.

; Acyl ester 1
; -----

[ chiral ]
x CD OE CE CC

[ trans ]
OF C1A CD x

[ out ]
OE C1A OF C1B

[ trans ]
C1B C1A OE CD

[ out ]
OF C1A OE C1B

; Acyl ester 2
; -----

[ out ]
y CE CD OG

[ chiral ]
z CE OG CD y

[ trans ]
OH C2A CE z

[ out ]
OG C2A OH C2B

[ trans ]
C2B C2A OG CE

[ out ]
OH C2A OG C2B
__MAP__

# dope.gromos.map
cat << __MAP__ > dope.gromos.map
; Created by TAW on 17.5.2013
[ molecule ]
DOPE

[ martini ]
NH3 PO4 GL1 GL2 C1A C2A D3A C4A C5A C1B C2B D3B C4B C5B

;
; NH3-PO4-GL1-C1A-C2A-D3A-C4A-C5A
; |
; GL2-C1B-C2B-D3B-C4B-C5A

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
; Terminal head group (choline)
1 H1 NH3
2 H2
3 H3
4 NTM
5 CA NH3 NH3 PO4
6 CB NH3 PO4
; Phosphate group
7 OA PO4 PO4 NH3
8 P PO4
9 OB
10 OC
11 OD PO4 PO4 GL1
; Diacylglycerol
12 CC GL1 GL1 PO4
13 CD GL1

```

```

14   OE   GL1 GL1 GL1 C1A
15   C1A   GL1 C1A
16   OF
17   C1B   C1A C1A C1A GL1
18   C1C   C1A
19   C1D   C1A C1A C2A
20   C1E   C1A C2A
21   C1F   C2A C2A C1A
22   C1G   C2A
23   C1H   C2A D3A
24   C1I   D3A D3A C2A
25   C1J   D3A D3A C4A
26   C1K   D3A C4A
27   C1L   C4A
28   C1M   C4A C4A C4A C5A
29   C1N   C4A C4A C5A
30   C1O   C5A C4A
31   C1P   C5A C5A C4A
32   C1Q   C5A C5A C5A C4A
33   C1R   C5A
34   CE   GL2
35   OG   GL2 GL2 GL2 C1B
36   C2A   GL2 C1B
37   OH
38   C2B   C1B C1B C1B GL2
39   C2C   C1B
40   C2D   C1B C1B C1B C2B
41   C2E   C1B C2B
42   C2F   C2B C2B C2B C1B
43   C2G   C2B
44   C2H   C2B D3B
45   C2I   D3B D3B C2B
46   C2J   D3B D3B C4B
47   C2K   D3B C4B
48   C2L   C4B
49   C2M   C4B C4B C4B C5B
50   C2N   C4B C4B C5B
51   C2O   C5B C4B
52   C2P   C5B C5B C4B
53   C2Q   C5B C5B C5B C4B
54   C2R   C5B

; Choline group
[out]
;CN2 NTM CN1 CA
[chiral]
;CN3 NTM CN1 CN2 CA

; Cis double bonds
[ cis ]
C1H C1I C1J C1K
C2H C2I C2J C2K

; Acyl esters
; =====
; This reconstruction is somewhat complex. Unfortunately
; the Gromos united atom force field does not have
; correct dihedrals for acyl esters and these groups
; have to be built with correct geometry. Only setting
; the C-O-CO-C dihedrals correct is not sufficient, as
; the distortions may be so large that the dihedrals
; are pushed over the barrier. Therefore, the whole
; glycerol group is rebuilt so as to form a buffer.

; Acyl ester 1
; -----

[ chiral ]
x CD OE CE CC

[ trans ]
OF C1A CD x

[ out ]
OE C1A OF C1B

[ trans ]
C1B C1A OE CD

[ out ]
OF C1A OE C1B

; Acyl ester 2
; -----

[ out ]
y CE CD OG

[ chiral ]
z CE OG CD y

[ trans ]
OH C2A CE z

[ out ]
OG C2A OH C2B

[ trans ]
C2B C2A OG CE

[ out ]
OH C2A OG C2B
__MAP__

# dppc.amber.map
cat << __MAP__ > dppc.amber.map
; Slipids by Joakim P. M. Jämbäck made for combining with Amber force fields

[ molecule ]
DPPC

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B C3B C4B
;
; NC3-PO4-GL1-C1A-C2A-C3A-C4A
; |
; GL2-C1B-C2B-C3B-C4B

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
; Terminal head group (choline)
1   N   NC3
2   C13  NC3

```



```

3 H13A NC3
4 H13B NC3
5 H13C NC3
6 C14 NC3
7 H14A NC3
8 H14B NC3
9 H14C NC3
10 C15 NC3
11 H15A NC3
12 H15B NC3
13 H15C NC3
14 C12 NC3 NC3 NC3 P04
15 H12A NC3 NC3 NC3 P04
16 H12B NC3 NC3 NC3 P04
17 C11 NC3 P04
18 H11A NC3 P04
19 H11B NC3 P04
; Phosphate group
20 P P04
21 O13 P04
22 O14 P04
23 O11 P04 P04 GL1
24 O12 P04 P04 P04 NC3
; Diacylglycerol
25 C1 GL1 GL1 P04
26 HA GL1 GL1 P04
27 HB GL1 GL1 P04
28 C2 GL1 GL1 GL2
29 HS GL1 GL1 GL2
30 O21 GL1 GL1 GL2 C1A
31 C21 GL1 C1A
32 O22 GL1
33 C22 C1A C1A GL1
34 H2R C1A C1A GL1
35 H2S C1A C1A GL1
36 C3 GL2 GL2 GL2 P04
37 HX GL2 GL2 GL2 P04
38 HY GL2 GL2 GL2 P04
39 O31 GL2
40 C31 GL2 GL2 C1B
41 O32 GL2
42 C32 C1B C1B GL2
43 H2X C1B C1B GL2
44 H2Y C1B C1B GL2
45 C23 C1A
46 H3R C1A
47 H3S C1A
48 C24 C1A C1A C2A
49 H4R C1A C1A C2A
50 H4S C1A C1A C2A
51 C25 C1A C2A
52 H5R C1A C2A
53 H5S C1A C2A
54 C26 C2A C2A C1A
55 H6R C2A C2A C1A
56 H6S C2A C2A C1A
57 C27 C2A
58 H7R C2A
59 H7S C2A
60 C28 C2A C2A C2A C3A
61 H8R C2A C2A C2A C3A
62 H8S C2A C2A C2A C3A
63 C29 C2A C2A C3A
64 H9R C2A C2A C3A
65 H9S C2A C2A C3A
66 C210 C3A C2A
67 H10R C3A C2A
68 H10S C3A C2A
69 C211 C3A C3A C2A
70 H11R C3A C3A C2A
71 H11S C3A C3A C2A
72 C212 C3A
73 H12R C3A
74 H12S C3A
75 C213 C3A C3A C4A
76 H13R C3A C3A C4A
77 H13S C3A C3A C4A
78 C214 C4A C3A
79 H14R C4A C3A
80 H14S C4A C3A
81 C215 C4A C4A C3A
82 H15R C4A C4A C3A
83 H15S C4A C4A C3A
84 C216 C4A
85 H16R C4A
86 H16S C4A
87 H16T C4A
88 C33 C1B
89 H3X C1B
90 H3Y C1B
91 C34 C1B C1B C2B
92 H4X C1B C1B C2B
93 H4Y C1B C1B C2B
94 C35 C1B C2B
95 H5X C1B C2B
96 H5Y C1B C2B
97 C36 C2B C2B C1B
98 H6X C2B C2B C1B
99 H6Y C2B C2B C1B
100 C37 C2B
101 H7X C2B
102 H7Y C2B
103 C38 C2B C2B C2B C3B
104 H8X C2B C2B C2B C3B
105 H8Y C2B C2B C2B C3B
106 C39 C2B C2B C3B
107 H9X C2B C2B C3B
108 H9Y C2B C2B C3B
109 C310 C3B C2B
110 H10X C3B C2B
111 H10Y C3B C2B
112 C311 C3B C3B C2B
113 H11X C3B C3B C2B
114 H11Y C3B C3B C2B
115 C312 C3B
116 H12X C3B
117 H12Y C3B
118 C313 C3B C3B C4B
119 H13X C3B C3B C4B
120 H13Y C3B C3B C4B
121 C314 C4B C3B
122 H14X C4B C3B
123 H14Y C4B C3B
124 C315 C4B C4B C3B
125 H15X C4B C4B C3B
126 H15Y C4B C4B C3B
127 C316 C4B
128 H16X C4B
129 H16Y C4B

```

```

130 H16Z C4B

;;;making a choline group
[out]
C14 N C13 C12
H14A N C13 C12
H14B N C13 C12
H14C N C13 C12

[ chiral ]
C15 N C12 C13 C14
H15A N C12 C13 C14
H15B N C12 C13 C14
H15C N C12 C13 C14

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32
__MAP__

# dppc.charmm36.map
cat << __MAP__ > dppc.charmm36.map
; Mapping file created 28/02/2013 by Kri

[ molecule ]
DPPC

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B C3B C4B
;
; NC3-PO4-GL1-C1A-C2A-C3A-C4A
;
; GL2-C1B-C2B-C3B-C4B

[ mapping ]
charmm27 charmm36

[ atoms ]
; Terminal head group (choline)
1 N NC3
2 C13 NC3
3 H13A NC3
4 H13B NC3
5 H13C NC3
6 C14 NC3
7 H14A NC3
8 H14B NC3
9 H14C NC3
10 C15 NC3
11 H15A NC3
12 H15B NC3
13 H15C NC3
14 C12 NC3 NC3 NC3 PO4
15 H12A NC3 NC3 NC3 PO4
16 H12B NC3 NC3 NC3 PO4
17 C11 NC3 PO4
18 H11A NC3 PO4
19 H11B NC3 PO4
; Phosphate group
20 P PO4
21 O13 PO4
22 O14 PO4
23 O11 PO4 PO4 GL1
24 O12 PO4 PO4 PO4 NC3
; Diacylglycerol
25 C1 GL1 GL1 PO4
26 HA GL1 GL1 PO4
27 HB GL1 GL1 PO4
28 C2 GL1 GL1 GL2
29 HS GL1 GL1 GL2
30 O21 GL1 GL1 GL2 C1A
31 C21 GL1 C1A
32 O22 GL1
33 C22 C1A C1A GL1
34 H2R C1A C1A GL1
35 H2S C1A C1A GL1
36 C3 GL2 GL2 GL2 PO4
37 HX GL2 GL2 GL2 PO4
38 HY GL2 GL2 GL2 PO4
39 O31 GL2
40 C31 GL2 GL2 C1B
41 O32 GL2
42 C32 C1B C1B GL2
43 H2X C1B C1B GL2
44 H2Y C1B C1B GL2
45 C23 C1A
46 H3R C1A
47 H3S C1A
48 C24 C1A C1A C2A
49 H4R C1A C1A C2A
50 H4S C1A C1A C2A
51 C25 C1A C2A
52 H5R C1A C2A
53 H5S C1A C2A
54 C26 C2A C2A C1A
55 H6R C2A C2A C1A
56 H6S C2A C2A C1A
57 C27 C2A
58 H7R C2A
59 H7S C2A
60 C28 C2A C2A C2A C3A
61 H8R C2A C2A C2A C3A
62 H8S C2A C2A C2A C3A
63 C29 C2A C2A C3A
64 H9R C2A C2A C3A
65 H9S C2A C2A C3A
66 C210 C3A C2A
67 H10R C3A C2A
68 H10S C3A C2A
69 C211 C3A C3A C2A
70 H11R C3A C3A C2A
71 H11S C3A C3A C2A
72 C212 C3A
73 H12R C3A
74 H12S C3A
75 C213 C3A C3A C4A
76 H13R C3A C3A C4A
77 H13S C3A C3A C4A

```

```

78 C214 C4A C3A
79 H14R C4A C3A
80 H14S C4A C3A
81 C215 C4A C4A C3A
82 H15R C4A C4A C3A
83 H15S C4A C4A C3A
84 C216 C4A
85 H16R C4A
86 H16S C4A
87 H16T C4A
88 C33 C1B
89 H3X C1B
90 H3Y C1B
91 C34 C1B C1B C2B
92 H4X C1B C1B C2B
93 H4Y C1B C1B C2B
94 C35 C1B C2B
95 H5X C1B C2B
96 H5Y C1B C2B
97 C36 C2B C2B C1B
98 H6X C2B C2B C1B
99 H6Y C2B C2B C1B
100 C37 C2B
101 H7X C2B
102 H7Y C2B
103 C38 C2B C2B C2B C3B
104 H8X C2B C2B C2B C3B
105 H8Y C2B C2B C2B C3B
106 C39 C2B C2B C3B
107 H9X C2B C2B C3B
108 H9Y C2B C2B C3B
109 C310 C3B C2B
110 H10X C3B C2B
111 H10Y C3B C2B
112 C311 C3B C3B C2B
113 H11X C3B C3B C2B
114 H11Y C3B C3B C2B
115 C312 C3B
116 H12X C3B
117 H12Y C3B
118 C313 C3B C3B C4B
119 H13X C3B C3B C4B
120 H13Y C3B C3B C4B
121 C314 C4B C3B
122 H14X C4B C3B
123 H14Y C4B C3B
124 C315 C4B C4B C3B
125 H15X C4B C4B C3B
126 H15Y C4B C4B C3B
127 C316 C4B
128 H16X C4B
129 H16Y C4B
130 H16Z C4B

;;making a choline group
[out]
C14 N C13 C12
H14A N C13 C12
H14B N C13 C12
H14C N C13 C12

[ chiral ]
C15 N C12 C13 C14
H15A N C12 C13 C14
H15B N C12 C13 C14
H15C N C12 C13 C14

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32
__MAP__

# dppc.gromos.map
cat << __MAP__ > dppc.gromos.map
;Created by Kri on 07.03.2013

[ molecule ]
DPPC

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B C3B C4B
;
;NC3-PO4-GL1-C1A-C2A-C3A-C4A
;
; GL2-C1B-C2B-C3B-C4B

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
; Terminal head group (choline)
1 CN1 NC3
2 CN2 NC3
3 CN3 NC3
4 NTM NC3
5 CA NC3 NC3 PO4
6 CB NC3 PO4
; Phosphate group
7 OA PO4 PO4 NC3
8 P PO4
9 OB PO4
10 OC PO4
11 OD PO4 PO4 GL1
; Diacylglycerol
12 CC GL1 GL1 PO4
13 CD GL1
14 OE GL1 GL1 GL1 C1A
15 C1A GL1 C1A
16 OF
17 C1B C1A C1A C1A GL1
18 C1C C1A
19 C1D C1A C1A C2A
20 C1E C1A C2A
21 C1F C2A C2A C1A
22 C1G C2A
23 C1H C2A C2A C3A
24 C1I C2A C3A
25 C1J C3A C3A C2A

```

```

26 C1K C3A
27 C1L C3A C3A C4A
28 C1M C3A C4A
29 C1N C4A C4A C3A
30 C1O C4A C4A C4A C3A
31 C1P C4A
32 CE GL2
33 OG GL2 GL2 GL2 C1B
34 C2A GL2 C1B
35 OH
36 C2B C1B C1B C1B GL2
37 C2C C1B
38 C2D C1B C1B C2B
39 C2E C1B C2B
40 C2F C2B C2B C1B
41 C2G C2B
42 C2H C2B C2B C3B
43 C2I C2B C3B
44 C2J C3B C3B C2B
45 C2K C3B
46 C2L C3B C3B C4B
47 C2M C3B C4B
48 C2N C4B C4B C3B
49 C2O C4B C4B C4B C3B
50 C2P C4B

;;;making a choline group
[out]
CN2 NTM CN1 CA
[chiral]
CN3 NTM CN1 CN2 CA

; Acyl esters
; =====
; This reconstruction is somewhat complex. Unfortunately
; the Gromos united atom force field does not have
; correct dihedrals for acyl esters and these groups
; have to be built with correct geometry. Only setting
; the C-O-CO-C dihedrals correct is not sufficient, as
; the distortions may be so large that the dihedrals
; are pushed over the barrier. Therefore, the whole
; glycerol group is rebuilt so as to form a buffer.

; Acyl ester 1
; -----

[ chiral ]
x CD OE CE CC

[ trans ]
OF C1A CD x

[ out ]
OE C1A OF C1B

[ trans ]
C1B C1A OE CD

[ out ]
OF C1A OE C1B

; Acyl ester 2
; -----

[ out ]
y CE CD OG

[ chiral ]
z CE OG CD y

[ trans ]
OH C2A CE z

[ out ]
OG C2A OH C2B

[ trans ]
C2B C2A OG CE

[ out ]
OH C2A OG C2B
__MAP__

# gln.amber.map
cat << __MAP__ > gln.amber.map
[ molecule ]
GLN

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1 N BB
2 H BB
3 CA BB
4 HA BB
5 CB SC1 BB BB
6 HB1 SC1 BB BB
7 HB2 SC1 BB BB
8 CG SC1 SC1 BB
9 HG1 SC1 SC1 BB
10 HG2 SC1 SC1 BB
11 CD SC1
12 OE1 SC1
13 NE2 SC1
14 HE21 SC1
15 HE22 SC1
16 C BB
17 O BB

[ chiral ]
CB CA N C
HB1 CA N C
HB2 CA N C

[ chiral ]
HA CA N CB C ; L-Gln
; HA CA N C CB ; D-Gln
__MAP__

# gln.charmm36.map
cat << __MAP__ > gln.charmm36.map
[ molecule ]
GLN

```

```

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN     BB
3      CA     BB
4      HA     BB
5      CB     SC1 BB BB
6      HB1    SC1 BB BB
7      HB2    SC1 BB BB
8      CG     SC1 SC1 BB
9      HG1    SC1 SC1 BB
10     HG2    SC1 SC1 BB
11     CD     SC1
12     OE1    SC1
13     NE2    SC1
14     HE21   SC1
15     HE22   SC1
16     C      BB
17     O      BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Gln
; HA     CA      N      C      CB ; D-Gln
__MAP__

# gln.gromos.map
cat << __MAP__ > gln.gromos.map
[ molecule ]
GLN

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA     BB
4      CB     SC1 BB BB
5      CG     SC1 SC1 BB
6      CD     SC1
7      OE1    SC1
8      NE2    SC1
9      HE21   SC1
10     HE22   SC1
11     C      BB
12     O      BB

[ chiral ]
CB      CA      N      C
__MAP__

# glu.amber.map
cat << __MAP__ > glu.amber.map
[ molecule ]
GLU

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1      N      BB
2      H      BB
3      CA     BB
4      HA     BB
5      CB     SC1 BB BB
6      HB1    SC1 BB BB
7      HB2    SC1 BB BB
8      CG     SC1 SC1 BB
9      HG1    SC1 SC1 BB
10     HG2    SC1 SC1 BB
11     CD     SC1
12     OE1    SC1
13     OE2    SC1
14     HE2    SC1
15     C      BB
16     O      BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Glu
; HA     CA      N      C      CB ; D-Glu
__MAP__

# glu.charmm36.map
cat << __MAP__ > glu.charmm36.map
[ molecule ]
GLU

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN     BB
3      CA     BB
4      HA     BB
5      CB     SC1 BB BB
6      HB1    SC1 BB BB
7      HB2    SC1 BB BB
8      CG     SC1 SC1 BB
9      HG1    SC1 SC1 BB
10     HG2    SC1 SC1 BB

```

```

11    CD    SC1
12    OE1    SC1
13    OE2    SC1
14    HE2    SC1
15    C      BB
16    O      BB

[ chiral ]
CB    CA    N    C
HB1   CA    N    C
HB2   CA    N    C

[ chiral ]
HA    CA    N    CB    C ; L-Glu
; HA   CA    N    C    CB ; D-Glu
__MAP__

# glu.gromos.map
cat << __MAP__ > glu.gromos.map
[ molecule ]
GLU

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA      BB
4      CB    SC1 BB BB
5      CG    SC1 SC1 BB
6      CD    SC1
7      OE1    SC1
8      OE2    SC1
9      HE2    SC1
10     C      BB
11     O      BB

[ chiral ]
CB    CA    N    C
__MAP__

# gly.amber.map
cat << __MAP__ > gly.amber.map
[ molecule ]
GLY

[ martini ]
BB

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1      N      BB
2      H      BB
3      CA      BB
4      HA1     BB
5      HA2     BB
9      C      BB
10     O      BB
__MAP__

# gly.charmm36.map
cat << __MAP__ > gly.charmm36.map
[ molecule ]
GLY

[ martini ]
BB

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN     BB
3      CA      BB
4      HA1     BB
5      HA2     BB
9      C      BB
10     O      BB
__MAP__

# gly.gromos.map
cat << __MAP__ > gly.gromos.map
[ molecule ]
GLY

[ martini ]
BB

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA      BB
4      C      BB
5      O      BB
__MAP__

# heptane.gromos.map
cat << __MAP__ > heptane.gromos.map
[ molecule ]
HEP

[ martini ]
C1 C2

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      C101   C1
2      C102   C1 C1 C1 C1 C1 C2
3      C103   C1 C1 C2
4      C104   C1 C2
5      C105   C2 C2 C1
6      C106   C2 C2 C2 C2 C2 C1

```

```

7 C107 C2
__MAP__

# his.amber.map
cat << __MAP__ > his.amber.map
[ molecule ]
HIS

[ martini ]
BB SC1 SC2 SC3

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1 N BB
2 H BB
3 CA BB
4 HA BB
5 CB SC1 BB
6 HB1 SC1 BB
7 HB2 SC1 BB
8 CG SC1
9 ND1 SC3 SC1
10 HD1 SC3 SC1
11 CE1 SC3 SC2
12 HE1 SC3
13 NE2 SC2
14 HE2 SC2
15 CD2 SC2 SC1
16 HD2 SC2 SC1
17 C BB
18 O BB

[ chiral ]
CB CA N C
HB1 CA N C
HB2 CA N C

[ chiral ]
HA CA N CB C ; L-His
; HA CA N C CB ; D-His

[out]
HD1 ND1 CG NE2
HD2 CD2 CE1 CG
HE2 NE2 CE1 CD2
__MAP__

# his.charmm36.map
cat << __MAP__ > his.charmm36.map
[ molecule ]
HIS ;;;; is in fact HSP but if one of hydrogen on a nitrogen is missing in the input topology file, will become HSE or HSD automatically

[ martini ]
BB SC1 SC2 SC3

[ mapping ]
charmm27 charmm36

[ atoms ]
1 N BB
2 HN BB
3 CA BB
4 HA BB
5 CB SC1 BB
6 HB1 SC1 BB
7 HB2 SC1 BB
8 CD2 SC2 SC1
9 HD2 SC2 SC1
10 CG SC1
11 NE2 SC2
12 HE2 SC2
13 ND1 SC3 SC1
14 HD1 SC3 SC1
15 CE1 SC3 SC2
16 HE1 SC3
17 C BB
18 O BB

[ chiral ]
CB CA N C
HB1 CA N C
HB2 CA N C

[ chiral ]
HA CA N CB C ; L-His
; HA CA N C CB ; D-His

[out]
HD1 ND1 CG NE2
HD2 CD2 CE1 CG
HE2 NE2 CE1 CD2
__MAP__

# his.gromos.map
cat << __MAP__ > his.gromos.map
[ molecule ]
HIS

[ martini ]
BB SC1 SC2 SC3

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1 N BB
2 H BB
3 CA BB
4 CB SC1 BB
5 CG SC1
6 ND1 SC3 SC1
7 HD1 SC3 SC1
8 CD2 SC2 SC1
9 HD2 SC2 SC1
10 CE1 SC3 SC3 SC2
11 HE1 SC3
12 NE2 SC2
13 HE2 SC2
14 C BB
15 O BB

[ chiral ]
CB CA N C

```

```

[ out ]
HD1 ND1 CG NE2
HD2 CD2 CE1 CG
HE2 NE2 CE1 CD2
__MAP__

# ile.amber.map
cat << __MAP__ > ile.amber.map
[ molecule ]
ILE

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB      SC1 BB BB
  7      CG2     SC1 SC1 BB
  8      HG21    SC1 SC1 BB
  9      HG22    SC1 SC1 BB
 10      HG23    SC1 SC1 BB
 11      CG1     SC1 SC1 BB
 12      HG11    SC1 SC1 BB
 13      HG12    SC1 SC1 BB
 14      CD      SC1
 15      HD1     SC1
 16      HD2     SC1
 17      HD3     SC1
 18      C       BB
 19      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Ile
; HA     CA      N      C      CB ; D-Ile

[ out ]
CG2 CB CG1 CA
HG21 CB CG1 CA
HG22 CB CG1 CA
HG23 CB CG1 CA

[ chiral ]
HB      CB      CA      CG2   CG1   ; 3S stereoisomer (natural form)
; HB     CB      CA      CG1   CG2   ; 3R stereoisomer
__MAP__

# ile.charmm36.map
cat << __MAP__ > ile.charmm36.map
[ molecule ]
ILE

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
  1      N      BB
  2      HN      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB      SC1 BB BB
  7      CG2     SC1 SC1 BB
  8      HG21    SC1 SC1 BB
  9      HG22    SC1 SC1 BB
 10      HG23    SC1 SC1 BB
 11      CG1     SC1 SC1 BB
 12      HG11    SC1 SC1 BB
 13      HG12    SC1 SC1 BB
 14      CD      SC1
 15      HD1     SC1
 16      HD2     SC1
 17      HD3     SC1
 18      C       BB
 19      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Ile
; HA     CA      N      C      CB ; D-Ile

[ out ]
CG2 CB CG1 CA
HG21 CB CG1 CA
HG22 CB CG1 CA
HG23 CB CG1 CA

[ chiral ]
HB      CB      CA      CG2   CG1   ; 3S stereoisomer (natural form)
; HB     CB      CA      CG1   CG2   ; 3R stereoisomer
__MAP__

# ile.gromos.map
cat << __MAP__ > ile.gromos.map
[ molecule ]
ILE

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      CB      SC1 BB BB

```



```

5   CG1   SC1 SC1 BB
6   CG2   SC1 SC1 BB
7   CD     SC1
8   C      BB
9   O      BB

[ chiral ]
CB   CA   N   C
__MAP__

# leu.amber.map
cat << __MAP__ > leu.amber.map
[ molecule ]
LEU

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1   N      BB
2   H      BB
3   CA     BB
4   HA     BB
5   CB     SC1 BB BB
6   HB1    SC1 BB BB
7   HB2    SC1 BB BB
8   CG     SC1 SC1 BB
9   HG     SC1 SC1 BB
10  CD1     SC1
11  HD11    SC1
12  HD12    SC1
13  HD13    SC1
14  CD2     SC1
15  HD21    SC1
16  HD22    SC1
17  HD23    SC1
18  C       BB
19  O       BB

[ chiral ]
CB   CA   N   C
HB1  CA   N   C
HB2  CA   N   C

[ chiral ]
HA   CA   N   CB   C ; L-Leu
; HA   CA   N   C   CB ; D-Leu

[ out ]
CD2  CG CD1 CB
HD21 CG CD1 CB
HD22 CG CD1 CB
HD23 CG CD1 CB
__MAP__

# leu.charmm36.map
cat << __MAP__ > leu.charmm36.map
[ molecule ]
LEU

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
1   N      BB
2   HN     BB
3   CA     BB
4   HA     BB
5   CB     SC1 BB BB
6   HB1    SC1 BB BB
7   HB2    SC1 BB BB
8   CG     SC1 SC1 BB
9   HG     SC1 SC1 BB
10  CD1     SC1
11  HD11    SC1
12  HD12    SC1
13  HD13    SC1
14  CD2     SC1
15  HD21    SC1
16  HD22    SC1
17  HD23    SC1
18  C       BB
19  O       BB

[ chiral ]
CB   CA   N   C
HB1  CA   N   C
HB2  CA   N   C

[ chiral ]
HA   CA   N   CB   C ; L-Leu
; HA   CA   N   C   CB ; D-Leu

[ out ]
CD2  CG CD1 CB
HD21 CG CD1 CB
HD22 CG CD1 CB
HD23 CG CD1 CB
__MAP__

# leu.gromos.map
cat << __MAP__ > leu.gromos.map
[ molecule ]
LEU

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1   N      BB
2   H      BB
3   CA     BB
4   CB     SC1 BB BB
5   CG     SC1 SC1 BB
6   CD1     SC1
7   CD2     SC1
8   C       BB

```

```

      9      O      BB

[ chiral ]
CB      CA      N      C
__MAP__

# lys.amber.map
cat << __MAP__ > lys.amber.map
[ molecule ]
LYS

[ martini ]
BB SC1 SC2

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      CG      SC1 SC1 BB
  9      HG1     SC1 SC1 BB
 10      HG2     SC1 SC1 BB
 11      CD      SC1
 12      HD1     SC1
 13      HD2     SC1
 14      CE      SC1 SC1 SC2
 15      HE1     SC1 SC1 SC2
 16      HE2     SC1 SC1 SC2
 17      NZ      SC2 SC2 SC1
 18      HZ1     SC2
 19      HZ2     SC2
 20      HZ3     SC2
 21      C       BB
 22      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
; HA      CA      N      CB      C ; L-Lys
; HA      CA      N      C      CB ; D-Lys
__MAP__

# lys.charmm36.map
cat << __MAP__ > lys.charmm36.map
[ molecule ]
LYS

[ martini ]
BB SC1 SC2

[ mapping ]
charmm27 charmm36

[ atoms ]
  1      N      BB
  2      HN      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      CG      SC1 SC1 BB
  9      HG1     SC1 SC1 BB
 10      HG2     SC1 SC1 BB
 11      CD      SC1
 12      HD1     SC1
 13      HD2     SC1
 14      CE      SC1 SC1 SC2
 15      HE1     SC1 SC1 SC2
 16      HE2     SC1 SC1 SC2
 17      NZ      SC2 SC2 SC1
 18      HZ1     SC2
 19      HZ2     SC2
 20      HZ3     SC2
 21      C       BB
 22      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
; HA      CA      N      CB      C ; L-Lys
; HA      CA      N      C      CB ; D-Lys
__MAP__

# lys.gromos.map
cat << __MAP__ > lys.gromos.map
[ molecule ]
LYS

[ martini ]
BB SC1 SC2

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      CB      SC1 BB BB
  5      CG      SC1 SC1 BB
  6      CD      SC1
  7      CE      SC1 SC1 SC2
  8      NZ      SC2 SC2 SC1
  9      HZ1     SC2
 10      HZ2     SC2
 11      HZ3     SC2
 12      C       BB
 13      O       BB

[ chiral ]
CB      CA      N      C
__MAP__

```

```

# met.amber.map
cat << __MAP__ > met.amber.map
[ molecule ]
MET

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      CG      SC1 BB
  9      HG1     SC1 BB
 10      HG2     SC1 BB
 11      SD      SC1 SC1 BB
 12      CE      SC1
 13      HE1     SC1
 14      HE2     SC1
 15      HE3     SC1
 16      C       BB
 17      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Met
; HA      CA      N      C      CB ; D-Met
__MAP__

# met.charmm36.map
cat << __MAP__ > met.charmm36.map
[ molecule ]
MET

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
  1      N      BB
  2      HN      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      CG      SC1 BB
  9      HG1     SC1 BB
 10      HG2     SC1 BB
 11      SD      SC1 SC1 BB
 12      CE      SC1
 13      HE1     SC1
 14      HE2     SC1
 15      HE3     SC1
 16      C       BB
 17      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Met
; HA      CA      N      C      CB ; D-Met
__MAP__

# met.gromos.map
cat << __MAP__ > met.gromos.map
[ molecule ]
MET

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      CB      SC1 BB BB
  5      CG      SC1 BB
  6      SD      SC1 SC1 BB
  7      CE      SC1
  8      C       BB
  9      O       BB

[ chiral ]
CB      CA      N      C
__MAP__

# phe.amber.map
cat << __MAP__ > phe.amber.map
[ molecule ]
PHE

[ martini ]
BB SC1 SC2 SC3

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB
  6      HB1     SC1 BB
  7      HB2     SC1 BB

```

```

      8      CG      SC1
      9      CD1      SC1 SC3
     10      HD1      SC1
     11      CE1      SC3
     12      HE1      SC3
     13      CZ      SC3 SC2
     14      HZ      SC3
     15      CE2      SC2
     16      HE2      SC2
     17      CD2      SC2 SC1
     18      HD2      SC2
     19      C        BB
     20      O        BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Phe
; HA     CA      N      C      CB ; D-Phe

[ out ]
; Add some helper points
P CD1 CD2 CZ
Q CD2 CD1 CZ
R CZ   CD1 CD2

[ trans ]
; Place hydrogens using helper points
HD1   P CE1   R
HD2   Q CE2   R
HE1 CE1   P   CG
HE2 CE2   Q   CG
HZ     R CE1   P

[ out ]
; Place ring carbons
CD1 CE1 HE1 R
CD2 CE2 HE2 R
CZ  CE1 HE1 P
__MAP__

# phe.charmm36.map
cat << __MAP__ > phe.charmm36.map
[ molecule ]
PHE

[ martini ]
BB SC1 SC2 SC3

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN     BB
3      CA      BB
4      HA      BB
5      CB      SC1 BB
6      HB1     SC1 BB
7      HB2     SC1 BB
8      CG      SC1
9      CD1      SC1 SC3
10     HD1      SC1
11     CE1      SC3
12     HE1      SC3
13     CZ      SC3 SC2
14     HZ      SC3
15     CD2      SC2 SC1
16     HD2      SC2
17     CE2      SC2
18     HE2      SC2
19     C        BB
20     O        BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Phe
; HA     CA      N      C      CB ; D-Phe

[ out ]
; Add some helper points
P CD1 CD2 CZ
Q CD2 CD1 CZ
R CZ   CD1 CD2

[ trans ]
; Place hydrogens using helper points
HD1   P CE1   R
HD2   Q CE2   R
HE1 CE1   P   CG
HE2 CE2   Q   CG
HZ     R CE1   P

[ out ]
; Place ring carbons
CD1 CE1 HE1 R
CD2 CE2 HE2 R
CZ  CE1 HE1 P
__MAP__

# phe.gromos.map
cat << __MAP__ > phe.gromos.map
[ molecule ]
PHE

[ martini ]
BB SC1 SC2 SC3

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA      BB
4      CB      SC1 BB
5      CG      SC1
6      CD1      SC1 SC3

```

```

7 HD1 SC1
8 CD2 SC2 SC1
9 HD2 SC2
10 CE1 SC3
11 HE1 SC3
12 CE2 SC2
13 HE2 SC2
14 CZ SC3 SC2
15 HZ SC3
16 C BB
17 O BB

[ chiral ]
CB CA N C

[ out ]
; Add some helper points
P CD1 CD2 CZ
Q CD2 CD1 CZ
R CZ CD1 CD2

[ trans ]
; Place hydrogens using helper points
HD1 F CE1 R
HD2 Q CE2 R
HE1 CE1 F CG
HE2 CE2 Q CG
HZ R CE1 P

[ out ]
; Place ring carbons
CD1 CE1 HE1 R
CD2 CE2 HE2 R
CZ CE1 HE1 P
__MAP__

# popc.amber.map
cat << __MAP__ > popc.amber.map
; Slipids by Joakim P. M. Jämbäck made for combining with Amber force field
[ molecule ]
POPC

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1 N NC3
2 C12 NC3 NC3 NC3 PO4
3 C13 NC3
4 C14 NC3
5 C15 NC3
6 H12A NC3 NC3 NC3 PO4
7 H12B NC3 NC3 NC3 PO4
8 H13A NC3
9 H13B NC3
10 H13C NC3
11 H14A NC3
12 H14B NC3
13 H14C NC3
14 H15A NC3
15 H15B NC3
16 H15C NC3
17 C11 NC3 PO4
18 H11A NC3 PO4
19 H11B NC3 PO4
20 P PO4
21 O13 PO4
22 O14 PO4
23 O12 PO4 PO4 PO4 NC3
24 O11 PO4 PO4 GL1
25 C1 GL1 GL1 PO4
26 HA GL1 GL1 PO4
27 HB GL1 GL1 PO4
28 C2 GL1 GL1 GL2
29 HS GL1 GL1 GL2
30 O21 GL1 GL1 GL2 C1B
31 C21 GL1 C1B
32 O22 GL1
33 C22 C1B C1B GL1
34 H2R C1B C1B GL1
35 H2S C1B C1B GL1
36 C3 GL2 GL2 GL2 PO4
37 HX GL2 GL2 GL2 PO4
38 HY GL2 GL2 GL2 PO4
39 O31 GL2
40 C31 GL2 GL1 C1A
41 O32 GL2
42 C32 C1A C1A GL2
43 H2X C1A C1A GL2
44 H2Y C1A C1A GL2
45 C23 C1B
46 H3R C1B
47 H3S C1B
48 C24 C1B C1B C2B
49 H4R C1B C1B C2B
50 H4S C1B C1B C2B
51 C25 C1B C2B
52 H5R C1B C2B
53 H5S C1B C2B
54 C26 C2B C2B C1B
55 H6R C2B C2B C1B
56 H6S C2B C2B C1B
57 C27 C2B
58 H7R C2B
59 H7S C2B
60 C28 C2B D3B
61 H8R C2B D3B
62 H8S C2B D3B
63 C29 D3B D3B C2B
64 H91 D3B
65 C210 D3B D3B C4B
66 H101 D3B
67 C211 C4B D3B
68 H11R C4B D3B
69 H11S C4B D3B
70 C212 C4B
71 H12R C4B
72 H12S C4B
73 C213 C4B C4B C4B C5B
74 H13R C4B C4B C4B C5B
75 H13S C4B C4B C4B C5B
76 C214 C4B C4B C5B
77 H14R C4B C4B C5B
78 H14S C4B C4B C5B
79 C215 C5B C4B

```

```

80 H15R C5B C4B
81 H15S C5B C4B
82 C216 C5B C5B C4B
83 H16R C5B C5B C4B
84 H16S C5B C5B C4B
85 C217 C5B C5B C5B C4B
86 H17R C5B C5B C5B C4B
87 H17S C5B C5B C5B C4B
88 C218 C5B
89 H18R C5B
90 H18S C5B
91 H18T C5B
92 C33 C1A
93 H3X C1A
94 H3Y C1A
95 C34 C1A C1A C2A
96 H4X C1A C1A C2A
97 H4Y C1A C1A C2A
98 C35 C1A C2A
99 H5X C1A C2A
100 H5Y C1A C2A
101 C36 C2A C2A C1A
102 H6X C2A C2A C1A
103 H6Y C2A C2A C1A
104 C37 C2A
105 H7X C2A
106 H7Y C2A
107 C38 C2A C2A C2A C3A
108 H8X C2A C2A C2A C3A
109 H8Y C2A C2A C2A C3A
110 C39 C2A C2A C3A
111 H9X C2A C2A C3A
112 H9Y C2A C2A C3A
113 C310 C3A C2A
114 H10X C3A C2A
115 H10Y C3A C2A
116 C311 C3A C3A C2A
117 H11X C3A C3A C2A
118 H11Y C3A C3A C2A
119 C312 C3A
120 H12X C3A
121 H12Y C3A
122 C313 C3A C3A C4A
123 H13X C3A C3A C4A
124 H13Y C3A C3A C4A
125 C314 C4A C3A
126 H14X C4A C3A
127 H14Y C4A C3A
128 C315 C4A C4A C3A
129 H15X C4A C4A C3A
130 H15Y C4A C4A C3A
131 C316 C4A
132 H16X C4A
133 H16Y C4A
134 H16Z C4A

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipind chains
[ out ]
H9R C29 C28 C210
[ trans ]
H10R C210 C29 C28
[ out ]
C211 C210 C29 H10R

;;;making a choline group
[out]
C14 N C13 C12
H14A N C13 C12
H14B N C13 C12
H14C N C13 C12

[ chiral ]
C15 N C12 C13 C14
H15A N C12 C13 C14
H15B N C12 C13 C14
H15C N C12 C13 C14
__MAP__

# popc.charmm36.map
cat << __MAP__ > popc.charmm36.map
; latest revision kri 12.3.2013
[ molecule ]
POPC

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

[ mapping ]
charmm27 charmm36

[ atoms ]
1 N NC3
2 C12 NC3 NC3 NC3 PO4
3 C13 NC3
4 C14 NC3
5 C15 NC3
6 H12A NC3 NC3 NC3 PO4
7 H12B NC3 NC3 NC3 PO4
8 H13A NC3
9 H13B NC3
10 H13C NC3
11 H14A NC3
12 H14B NC3
13 H14C NC3
14 H15A NC3
15 H15B NC3
16 H15C NC3
17 C11 NC3 PO4
18 H11A NC3 PO4
19 H11B NC3 PO4
20 P PO4
21 O13 PO4
22 O14 PO4
23 O12 PO4 PO4 PO4 NC3

```

```

24 O11 PO4 PO4 GL1
25 C1 GL1 GL1 PO4
26 HA GL1 GL1 PO4
27 HB GL1 GL1 PO4
28 C2 GL1 GL1 GL2
29 HS GL1 GL1 GL2
30 O21 GL1 GL1 GL2 C1B
31 C21 GL1 C1B
32 O22 GL1
33 C22 C1B C1B GL1
34 H2R C1B C1B GL1
35 H2S C1B C1B GL1
36 C3 GL2 GL2 GL2 PO4
37 HX GL2 GL2 GL2 PO4
38 HY GL2 GL2 GL2 PO4
39 O31 GL2
40 C31 GL2 GL1 C1A
41 O32 GL2
42 C32 C1A C1A GL2
43 H2X C1A C1A GL2
44 H2Y C1A C1A GL2
45 C23 C1B
46 H3R C1B
47 H3S C1B
48 C24 C1B C1B C2B
49 H4R C1B C1B C2B
50 H4S C1B C1B C2B
51 C25 C1B C2B
52 H5R C1B C2B
53 H5S C1B C2B
54 C26 C2B C2B C1B
55 H6R C2B C2B C1B
56 H6S C2B C2B C1B
57 C27 C2B
58 H7R C2B
59 H7S C2B
60 C28 C2B D3B
61 H8R C2B D3B
62 H8S C2B D3B
63 C29 D3B D3B C2B
64 H91 D3B
65 C210 D3B D3B C4B
66 H101 D3B
67 C211 C4B D3B
68 H11R C4B D3B
69 H11S C4B D3B
70 C212 C4B
71 H12R C4B
72 H12S C4B
73 C213 C4B C4B C4B C5B
74 H13R C4B C4B C4B C5B
75 H13S C4B C4B C4B C5B
76 C214 C4B C4B C5B
77 H14R C4B C4B C5B
78 H14S C4B C4B C5B
79 C215 C5B C4B
80 H15R C5B C4B
81 H15S C5B C4B
82 C216 C5B C5B C4B
83 H16R C5B C5B C4B
84 H16S C5B C5B C4B
85 C217 C5B C5B C5B C4B
86 H17R C5B C5B C5B C4B
87 H17S C5B C5B C5B C4B
88 C218 C5B
89 H18R C5B
90 H18S C5B
91 H18T C5B
92 C33 C1A
93 H3X C1A
94 H3Y C1A
95 C34 C1A C1A C2A
96 H4X C1A C1A C2A
97 H4Y C1A C1A C2A
98 C35 C1A C2A
99 H5X C1A C2A
100 H5Y C1A C2A
101 C36 C2A C2A C1A
102 H6X C2A C2A C1A
103 H6Y C2A C2A C1A
104 C37 C2A
105 H7X C2A
106 H7Y C2A
107 C38 C2A C2A C2A C3A
108 H8X C2A C2A C2A C3A
109 H8Y C2A C2A C2A C3A
110 C39 C2A C2A C3A
111 H9X C2A C2A C3A
112 H9Y C2A C2A C3A
113 C310 C3A C2A
114 H10X C3A C2A
115 H10Y C3A C2A
116 C311 C3A C3A C2A
117 H11X C3A C3A C2A
118 H11Y C3A C3A C2A
119 C312 C3A
120 H12X C3A
121 H12Y C3A
122 C313 C3A C3A C4A
123 H13X C3A C3A C4A
124 H13Y C3A C3A C4A
125 C314 C4A C3A
126 H14X C4A C3A
127 H14Y C4A C3A
128 C315 C4A C4A C3A
129 H15X C4A C4A C3A
130 H15Y C4A C4A C3A
131 C316 C4A
132 H16X C4A
133 H16Y C4A
134 H16Z C4A

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipid chains
[ out ]
H9R C29 C28 C210

```

```

[ trans ]
H10R C210 C29 C28
[ out ]
C211 C210 C29 H10R

;;;making a choline group
[out]
C14 N C13 C12
H14A N C13 C12
H14B N C13 C12
H14C N C13 C12

[ chiral ]
C15 N C12 C13 C14
H15A N C12 C13 C14
H15B N C12 C13 C14
H15C N C12 C13 C14
__MAP__

# popc.gromos.map
cat << __MAP__ > popc.gromos.map
;Created by Kri on 06.03.2013
[ molecule ]
POPC

[ martini ]
NC3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

;
;NC3-PO4-GL1-C1B-C2B-D3B-C4B-C5B
;
;      |
;      GL2-C1A-C2A-C3A-C4A

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
; Terminal head group (choline)
1  CN1  NC3
2  CN2
3  CN3
4  NTM
5  CA  NC3 NC3 PO4
6  CB  NC3 PO4
; Phosphate group
7  OA  PO4 PO4 NC3
8  P   PO4
9  OB
10 OC
11 OD  PO4 PO4 GL1
; Diacylglycerol
12 CC  GL1 GL1 PO4
13 CD  GL1
14 OE  GL1 GL1 GL1 C1B
15 C1A GL1 C1B
16 OF
17 C1B  C1B C1B C1B GL1
18 C1C  C1B
19 C1D  C1B C1B C2B
20 C1E  C1B C2B
21 C1F  C2B C2B C1B
22 C1G  C2B
23 C1H  C2B D3B
24 C1I  D3B D3B C2B
25 C1J  D3B D3B C4B
26 C1K  D3B C4B
27 C1L  C4B
28 C1M  C4B C4B C4B C5B
29 C1N  C4B C4B C5B
30 C1O  C5B C4B
31 C1P  C5B C5B C4B
32 C1Q  C5B C5B C5B C4B
33 C1R  C5B
34 CE  GL2
35 OG  GL2 GL2 GL2 C1A
36 C2A GL2 C1A
37 OH
38 C2B  C1A C1A C1A GL2
39 C2C  C1A
40 C2D  C1A C1A C1A C2A
41 C2E  C1A C2A
42 C2F  C2A C2A C2A C1A
43 C2G  C2A
44 C2H  C2A C2A C2A C3A
45 C2I  C2A C3A
46 C2J  C3A C3A C3A C2A
47 C2K  C3A
48 C2L  C3A C3A C3A C4A
49 C2M  C3A C4A
50 C2N  C4A C4A C3A
51 C2O  C4A C4A C4A C3A
52 C2P  C4A

;;;making a choline group
[out]
;CN2 NTM CN1 CA
[chiral]
;CN3 NTM CN1 CN2 CA

[ cis ]
;cis double bond
C1H C1I C1J C1K

; Acyl esters
; =====
; This reconstruction is somewhat complex. Unfortunately
; the Gromos united atom force field does not have
; correct dihedrals for acyl esters and these groups
; have to be built with correct geometry. Only setting
; the C-O-CO-C dihedrals correct is not sufficient, as
; the distortions may be so large that the dihedrals
; are pushed over the barrier. Therefore, the whole
; glycerol group is rebuilt so as to form a buffer.

; Acyl ester 1
; -----

[ chiral ]
x CD OE CE CC

[ trans ]
OF C1A CD x

[ out ]
OE C1A OF C1B

[ trans ]

```



```

C1B C1A OE CD

[ out ]
OF C1A OE C1B

; Acyl ester 2
; -----

[ out ]
y CE CD OG

[ chiral ]
z CE OG CD y

[ trans ]
OH C2A CE z

[ out ]
OG C2A OH C2B

[ trans ]
C2B C2A OG CE

[ out ]
OH C2A OG C2B
__MAP__

# pope.amber.map
cat << __MAP__ > pope.amber.map
; Slipids by Joakim P. M. Jämbäck made for combining with Amber force field

[ molecule ]
POPE

[ martini ]
NH3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B
;
; NH3-PO4-GL1-C1B-C2B-D3B-C4B-C5B
; |
; GL2-C1A-C2A-C3A-C4A

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N   NH3
  2  HN1   NH3
  3  HN2   NH3
  4  HN3   NH3
  5   C12   NH3 NH3 PO4
  6  H12A   NH3 NH3 PO4
  7  H12B   NH3 NH3 PO4
  8   C11   NH3 PO4 PO4
  9  H11A   NH3 PO4 PO4
 10  H11B   NH3 PO4 PO4
 11      P   PO4
 12   O13   PO4
 13   O14   PO4
 14   O12   PO4 PO4 PO4 NH3
 15   O11   PO4 PO4 GL1
 16    C1   GL1 GL1 PO4
 17   HA   GL1 GL1 PO4
 18   HB   GL1 GL1 PO4
 19    C2   GL1 GL1 GL2
 20   HS   GL1 GL1 GL2
 21   O21   GL1 GL1 GL2 C1B
 22   C21   GL1 C1B
 23   O22   GL1
 24   C22   C1B C1B GL1
 25   H2R   C1B C1B GL1
 26   H2S   C1B C1B GL1
 27    C3   GL2 GL2 GL2 PO4
 28   HX   GL2 GL2 GL2 PO4
 29   HY   GL2 GL2 GL2 PO4
 30   O31   GL2
 31   C31   GL2 GL1 C1A
 32   O32   GL2
 33   C32   C1A C1A GL2
 34   H2X   C1A C1A GL2
 35   H2Y   C1A C1A GL2
 36   C23   C1B
 37   H3R   C1B
 38   H3S   C1B
 39   C24   C1B C1B C2B
 40   H4R   C1B C1B C2B
 41   H4S   C1B C1B C2B
 42   C25   C1B C2B
 43   H5R   C1B C2B
 44   H5S   C1B C2B
 45   C26   C2B C2B C1B
 46   H6R   C2B C2B C1B
 47   H6S   C2B C2B C1B
 48   C27   C2B
 49   H7R   C2B
 50   H7S   C2B
 51   C28   C2B D3B
 52   H8R   C2B D3B
 53   H8S   C2B D3B
 54   C29   D3B D3B C2B
 55   H91   D3B
 56   C210  D3B D3B C4B
 57   H101  D3B
 58   C211  C4B D3B
 59   H11R  C4B D3B
 60   H11S  C4B D3B
 61   C212  C4B
 62   H12R  C4B
 63   H12S  C4B
 64   C213  C4B C4B C4B C5B
 65   H13R  C4B C4B C4B C5B
 66   H13S  C4B C4B C4B C5B
 67   C214  C4B C4B C5B
 68   H14R  C4B C4B C5B
 69   H14S  C4B C4B C5B
 70   C215  C5B C4B
 71   H15R  C5B C4B
 72   H15S  C5B C4B
 73   C216  C5B C5B C4B
 74   H16R  C5B C5B C4B
 75   H16S  C5B C5B C4B
 76   C217  C5B C5B C5B C4B
 77   H17R  C5B C5B C5B C4B
 78   H17S  C5B C5B C5B C4B
 79   C218  C5B
 80   H18R  C5B
 81   H18S  C5B
 82   H18T  C5B

```

```

83 C33 C1A
84 H3X C1A
85 H3Y C1A
86 C34 C1A C1A C2A
87 H4X C1A C1A C2A
88 H4Y C1A C1A C2A
89 C35 C1A C2A
90 H5X C1A C2A
91 H5Y C1A C2A
92 C36 C2A C2A C1A
93 H6X C2A C2A C1A
94 H6Y C2A C2A C1A
95 C37 C2A
96 H7X C2A
97 H7Y C2A
98 C38 C2A C2A C2A C3A
99 H8X C2A C2A C2A C3A
100 H8Y C2A C2A C2A C3A
101 C39 C2A C2A C3A
102 H9X C2A C2A C3A
103 H9Y C2A C2A C3A
104 C310 C3A C2A
105 H10X C3A C2A
106 H10Y C3A C2A
107 C311 C3A C3A C2A
108 H11X C3A C3A C2A
109 H11Y C3A C3A C2A
110 C312 C3A
111 H12X C3A
112 H12Y C3A
113 C313 C3A C3A C4A
114 H13X C3A C3A C4A
115 H13Y C3A C3A C4A
116 C314 C4A C3A
117 H14X C4A C3A
118 H14Y C4A C3A
119 C315 C4A C4A C3A
120 H15X C4A C4A C3A
121 H15Y C4A C4A C3A
122 C316 C4A
123 H16X C4A
124 H16Y C4A
125 H16Z C4A

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipind chains
[ out ]
H91 C29 C28 C210
[ trans ]
H101 C210 C29 C28
[ out ]
C211 C210 C29 H101
__MAP__

# pope.charmm36.map
cat << __MAP__ > pope.charmm36.map
; latest revision kri 15.3.2013
[ molecule ]
POPE

[ martini ]
NH3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B
;
; NH3-PO4-GL1-C1B-C2B-D3B-C4B-C5B
;
; GL2-C1A-C2A-C3A-C4A

[ mapping ]
charmm27 charmm36

[ atoms ]
1 N NH3
2 HN1 NH3
3 HN2 NH3
4 HN3 NH3
5 C12 NH3 NH3 PO4
6 H12A NH3 NH3 PO4
7 H12B NH3 NH3 PO4
8 C11 NH3 PO4 PO4
9 H11A NH3 PO4 PO4
10 H11B NH3 PO4 PO4
11 P PO4
12 O13 PO4
13 O14 PO4
14 O12 PO4 PO4 PO4 NH3
15 O11 PO4 PO4 GL1
16 C1 GL1 GL1 PO4
17 HA GL1 GL1 PO4
18 HB GL1 GL1 PO4
19 C2 GL1 GL1 GL2
20 HS GL1 GL1 GL2
21 O21 GL1 GL1 GL2 C1B
22 C21 GL1 C1B
23 O22 GL1
24 C22 C1B C1B GL1
25 H2R C1B C1B GL1
26 H2S C1B C1B GL1
27 C3 GL2 GL2 GL2 PO4
28 HX GL2 GL2 GL2 PO4
29 HY GL2 GL2 GL2 PO4
30 O31 GL2
31 C31 GL2 GL1 C1A
32 O32 GL2
33 C32 C1A C1A GL2
34 H2X C1A C1A GL2
35 H2Y C1A C1A GL2
36 C23 C1B
37 H3R C1B
38 H3S C1B
39 C24 C1B C1B C2B
40 H4R C1B C1B C2B
41 H4S C1B C1B C2B
42 C25 C1B C2B
43 H5R C1B C2B

```

```

44 H5S C1B C2B
45 C26 C2B C2B C1B
46 H6R C2B C2B C1B
47 H6S C2B C2B C1B
48 C27 C2B
49 H7R C2B
50 H7S C2B
51 C28 C2B D3B
52 H8R C2B D3B
53 H8S C2B D3B
54 C29 D3B D3B C2B
55 H91 D3B
56 C210 D3B D3B C4B
57 H101 D3B
58 C211 C4B D3B
59 H11R C4B D3B
60 H11S C4B D3B
61 C212 C4B
62 H12R C4B
63 H12S C4B
64 C213 C4B C4B C4B C5B
65 H13R C4B C4B C4B C5B
66 H13S C4B C4B C4B C5B
67 C214 C4B C4B C5B
68 H14R C4B C4B C5B
69 H14S C4B C4B C5B
70 C215 C5B C4B
71 H15R C5B C4B
72 H15S C5B C4B
73 C216 C5B C5B C4B
74 H16R C5B C5B C4B
75 H16S C5B C5B C4B
76 C217 C5B C5B C5B C4B
77 H17R C5B C5B C5B C4B
78 H17S C5B C5B C5B C4B
79 C218 C5B
80 H18R C5B
81 H18S C5B
82 H18T C5B
83 C33 C1A
84 H3X C1A
85 H3Y C1A
86 C34 C1A C1A C2A
87 H4X C1A C1A C2A
88 H4Y C1A C1A C2A
89 C35 C1A C2A
90 H5X C1A C2A
91 H5Y C1A C2A
92 C36 C2A C2A C1A
93 H6X C2A C2A C1A
94 H6Y C2A C2A C1A
95 C37 C2A
96 H7X C2A
97 H7Y C2A
98 C38 C2A C2A C2A C3A
99 H8X C2A C2A C2A C3A
100 H8Y C2A C2A C2A C3A
101 C39 C2A C2A C3A
102 H9X C2A C2A C3A
103 H9Y C2A C2A C3A
104 C310 C3A C2A
105 H10X C3A C2A
106 H10Y C3A C2A
107 C311 C3A C3A C2A
108 H11X C3A C3A C2A
109 H11Y C3A C3A C2A
110 C312 C3A
111 H12X C3A
112 H12Y C3A
113 C313 C3A C3A C4A
114 H13X C3A C3A C4A
115 H13Y C3A C3A C4A
116 C314 C4A C3A
117 H14X C4A C3A
118 H14Y C4A C3A
119 C315 C4A C4A C3A
120 H15X C4A C4A C3A
121 H15Y C4A C4A C3A
122 C316 C4A
123 H16X C4A
124 H16Y C4A
125 H16Z C4A

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipind chains
[ out ]
H91 C29 C28 C210
[ trans ]
H101 C210 C29 C28
[ out ]
C211 C210 C29 H101
__MAP__

# pope.gromos.map
cat << __MAP__ > pope.gromos.map
; Mapping file created 11/02/2013 by TAW

[ molecule ]
POPE

[ martini ]
NH3 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
; Terminal head group (ethanolamin)
1 H1 NH3
2 H2
3 H3
4 NTM
5 CA NH3 NH3 PO4
6 CB NH3 PO4
; Phosphate group

```

```

7   OA   PO4 PO4 NH3
8   P    PO4
9   OB
10  OC
11  OD   PO4 PO4 GL1
; Diacylglycerol
12  CC   GL1 GL1 PO4
13  CD   GL1
14  OE   GL1 GL1 GL1 C1B
15  C1A  GL1 C1B
16  OF
17  C1B  C1B C1B C1B GL1
18  C1C  C1B
19  C1D  C1B C1B C2B
20  C1E  C1B C2B
21  C1F  C2B C2B C1B
22  C1G  C2B
23  C1H  C2B D3B
24  C1I  D3B D3B C2B
25  C1J  D3B D3B C4B
26  C1K  D3B C4B
27  C1L  C4B
28  C1M  C4B C4B C4B C5B
29  C1N  C4B C4B C5B
30  C1O  C5B C4B
31  C1P  C5B C5B C4B
32  C1Q  C5B C5B C5B C4B
33  C1R  C5B
34  CE   GL2
35  OG   GL2 GL2 GL2 C1A
36  C2A  GL2 C1A
37  OH
38  C2B  C1A C1A C1A GL2
39  C2C  C1A
40  C2D  C1A C1A C1A C2A
41  C2E  C1A C2A
42  C2F  C2A C2A C1A
43  C2G  C2A
44  C2H  C2A C2A C3A
45  C2I  C2A C3A
46  C2J  C3A C3A C2A
47  C2K  C3A
48  C2L  C3A C3A C4A
49  C2M  C3A C4A
50  C2N  C4A C4A C3A
51  C2O  C4A C4A C4A C3A
52  C2P  C4A
__MAP__

# popg.amber.map
cat << __MAP__ > popg.amber.map
; Slipids by Joakim P. M. Jämbäck made for combining with Amber force field
[ molecule ]
POPG

[ martini ]
GL0 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1   C13   GL0
2   H13A  GL0
3   H13B  GL0
4   OC3   GL0
5   HO3   GL0
6   C12   GL0 GL0 GL0 PO4
7   H12A  GL0
8   OC2   GL0
9   HO2   GL0
17  C11   GL0 PO4
18  H11A  GL0 PO4
19  H11B  GL0 PO4
20  P     PO4
21  O13   PO4
22  O14   PO4
23  O12   PO4 PO4 PO4 GL0
24  O11   PO4 PO4 GL1
25  C1    GL1 GL1 PO4
26  HA    GL1 GL1 PO4
27  HB    GL1 GL1 PO4
28  C2    GL1 GL1 GL2
29  HS    GL1 GL1 GL2
30  O21   GL1 GL1 GL2 C1B
31  C21   GL1 C1B
32  O22   GL1
33  C22   C1B C1B GL1
34  H2R   C1B C1B GL1
35  H2S   C1B C1B GL1
36  C3    GL2 GL2 GL2 PO4
37  HX    GL2 GL2 GL2 PO4
38  HY    GL2 GL2 GL2 PO4
39  O31   GL2
40  C31   GL2 GL1 C1A
41  O32   GL2
42  C32   C1A C1A GL2
43  H2X   C1A C1A GL2
44  H2Y   C1A C1A GL2
45  C23   C1B
46  H3R   C1B
47  H3S   C1B
48  C24   C1B C1B C2B
49  H4R   C1B C1B C2B
50  H4S   C1B C1B C2B
51  C25   C1B C2B
52  H5R   C1B C2B
53  H5S   C1B C2B
54  C26   C2B C2B C1B
55  H6R   C2B C2B C1B
56  H6S   C2B C2B C1B
57  C27   C2B
58  H7R   C2B
59  H7S   C2B
60  C28   C2B D3B
61  H8R   C2B D3B
62  H8S   C2B D3B
63  C29   D3B D3B C2B
64  H91   D3B
65  C210  D3B D3B C4B
66  H101  D3B
67  C211  C4B D3B
68  H11R  C4B D3B
69  H11S  C4B D3B
70  C212  C4B
71  H12R  C4B
72  H12S  C4B
73  C213  C4B C4B C4B C5B

```

```

74 H13R C4B C4B C4B C5B
75 H13S C4B C4B C4B C5B
76 C214 C4B C4B C5B
77 H14R C4B C4B C5B
78 H14S C4B C4B C5B
79 C215 C5B C4B
80 H15R C5B C4B
81 H15S C5B C4B
82 C216 C5B C5B C4B
83 H16R C5B C5B C4B
84 H16S C5B C5B C4B
85 C217 C5B C5B C5B C4B
86 H17R C5B C5B C5B C4B
87 H17S C5B C5B C5B C4B
88 C218 C5B
89 H18R C5B
90 H18S C5B
91 H18T C5B
92 C33 C1A
93 H3X C1A
94 H3Y C1A
95 C34 C1A C1A C2A
96 H4X C1A C1A C2A
97 H4Y C1A C1A C2A
98 C35 C1A C2A
99 H5X C1A C2A
100 H5Y C1A C2A
101 C36 C2A C2A C1A
102 H6X C2A C2A C1A
103 H6Y C2A C2A C1A
104 C37 C2A
105 H7X C2A
106 H7Y C2A
107 C38 C2A C2A C2A C3A
108 H8X C2A C2A C2A C3A
109 H8Y C2A C2A C2A C3A
110 C39 C2A C2A C3A
111 H9X C2A C2A C3A
112 H9Y C2A C2A C3A
113 C310 C3A C2A
114 H10X C3A C2A
115 H10Y C3A C2A
116 C311 C3A C3A C2A
117 H11X C3A C3A C2A
118 H11Y C3A C3A C2A
119 C312 C3A
120 H12X C3A
121 H12Y C3A
122 C313 C3A C3A C4A
123 H13X C3A C3A C4A
124 H13Y C3A C3A C4A
125 C314 C4A C3A
126 H14X C4A C3A
127 H14Y C4A C3A
128 C315 C4A C4A C3A
129 H15X C4A C4A C3A
130 H15Y C4A C4A C3A
131 C316 C4A
132 H16X C4A
133 H16Y C4A
134 H16Z C4A

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipid chains
[ out ]
H91 C29 C28 C210
[ trans ]
H101 C210 C29 C28
[ out ]
C211 C210 C29 H101

;;;making a glycerol headgroup
[trans]
HO3 OC3 C13 C12
[out]
OC2 C12 HO3 C11
[chiral]
H12A C12 OC2 C13 C11
OC2 C12 C11 C13 H12A
HO2 C12 C11 C13 H12A
__MAP__

# popg.charmm36.map
cat << __MAP__ > popg.charmm36.map
; latest revision kri 15.3.2013
[ molecule ]
POPG

[ martini ]
GL0 PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

[ mapping ]
charmm27 charmm36

[ atoms ]
1 C13 GL0
2 H13A GL0
3 H13B GL0
4 OC3 GL0
5 HO3 GL0
6 C12 GL0 GL0 GL0 PO4
7 H12A GL0
8 OC2 GL0
9 HO2 GL0
17 C11 GL0 PO4
18 H11A GL0 PO4
19 H11B GL0 PO4
20 P PO4
21 O13 PO4
22 O14 PO4
23 O12 PO4 PO4 PO4 GL0
24 O11 PO4 PO4 GL1
25 C1 GL1 GL1 PO4
26 HA GL1 GL1 PO4
27 HB GL1 GL1 PO4

```

```

28 C2 GL1 GL1 GL2
29 HS GL1 GL1 GL2
30 O21 GL1 GL1 GL2 C1B
31 C21 GL1 C1B
32 O22 GL1
33 C22 C1B C1B GL1
34 H2R C1B C1B GL1
35 H2S C1B C1B GL1
36 C3 GL2 GL2 GL2 PO4
37 HX GL2 GL2 GL2 PO4
38 HY GL2 GL2 GL2 PO4
39 O31 GL2
40 C31 GL2 GL1 C1A
41 O32 GL2
42 C32 C1A C1A GL2
43 H2X C1A C1A GL2
44 H2Y C1A C1A GL2
45 C23 C1B
46 H3R C1B
47 H3S C1B
48 C24 C1B C1B C2B
49 H4R C1B C1B C2B
50 H4S C1B C1B C2B
51 C25 C1B C2B
52 H5R C1B C2B
53 H5S C1B C2B
54 C26 C2B C2B C1B
55 H6R C2B C2B C1B
56 H6S C2B C2B C1B
57 C27 C2B
58 H7R C2B
59 H7S C2B
60 C28 C2B D3B
61 H8R C2B D3B
62 H8S C2B D3B
63 C29 D3B D3B C2B
64 H91 D3B
65 C210 D3B D3B C4B
66 H101 D3B
67 C211 C4B D3B
68 H11R C4B D3B
69 H11S C4B D3B
70 C212 C4B
71 H12R C4B
72 H12S C4B
73 C213 C4B C4B C4B C5B
74 H13R C4B C4B C4B C5B
75 H13S C4B C4B C4B C5B
76 C214 C4B C4B C5B
77 H14R C4B C4B C5B
78 H14S C4B C4B C5B
79 C215 C5B C4B
80 H15R C5B C4B
81 H15S C5B C4B
82 C216 C5B C5B C4B
83 H16R C5B C5B C4B
84 H16S C5B C5B C4B
85 C217 C5B C5B C5B C4B
86 H17R C5B C5B C5B C4B
87 H17S C5B C5B C5B C4B
88 C218 C5B
89 H18R C5B
90 H18S C5B
91 H18T C5B
92 C33 C1A
93 H3X C1A
94 H3Y C1A
95 C34 C1A C1A C2A
96 H4X C1A C1A C2A
97 H4Y C1A C1A C2A
98 C35 C1A C2A
99 H5X C1A C2A
100 H5Y C1A C2A
101 C36 C2A C2A C1A
102 H6X C2A C2A C1A
103 H6Y C2A C2A C1A
104 C37 C2A
105 H7X C2A
106 H7Y C2A
107 C38 C2A C2A C2A C3A
108 H8X C2A C2A C2A C3A
109 H8Y C2A C2A C2A C3A
110 C39 C2A C2A C3A
111 H9X C2A C2A C3A
112 H9Y C2A C2A C3A
113 C310 C3A C2A
114 H10X C3A C2A
115 H10Y C3A C2A
116 C311 C3A C3A C2A
117 H11X C3A C3A C2A
118 H11Y C3A C3A C2A
119 C312 C3A
120 H12X C3A
121 H12Y C3A
122 C313 C3A C3A C4A
123 H13X C3A C3A C4A
124 H13Y C3A C3A C4A
125 C314 C4A C3A
126 H14X C4A C3A
127 H14Y C4A C3A
128 C315 C4A C4A C3A
129 H15X C4A C4A C3A
130 H15Y C4A C4A C3A
131 C316 C4A
132 H16X C4A
133 H16Y C4A
134 H16Z C4A

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipid chains
[ out ]
H91 C29 C28 C210
[ trans ]
H101 C210 C29 C28
[ out ]
C211 C210 C29 H101

```

```

;;;making a glycerol headgroup
[trans]
HO3 OC3 C13 C12
[out]
OC2 C12 HO3 C11
[chiral]
H12A C12 OC2 C13 C11
OC2 C12 C11 C13 H12A
HO2 C12 C11 C13 H12A
__MAP__

# pops.amber.map
cat << __MAP__ > pops.amber.map
; SlipIDs by Joakim P. M. Jämbbeck made for combining with Amber force field
[ molecule ]
POPS

[ martini ]
CNO PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1 N CNO
2 HN1 CNO
3 HN2 CNO
4 HN3 CNO
5 C12 CNO CNO CNO PO4
6 H12A CNO CNO CNO PO4
7 C13 CNO
8 O13A CNO
9 O13B CNO
10 C11 PO4 CNO
11 H11A PO4 CNO
12 H11B PO4 CNO
13 P PO4
14 O13
15 O14
16 O12
17 O11 PO4 PO4 GL1
18 C1 GL1 GL1 PO4
19 HA GL1 GL1 PO4
20 HB GL1 GL1 PO4
21 C2 GL1 GL2
22 HS GL1 GL2
23 O21 GL1 GL1 GL1 C1B
24 C21 GL1 C1B
25 O22 GL1
26 C22 C1B C1B GL1
27 H2R C1B C1B GL1
28 H2S C1B C1B GL1
29 C3 GL2 GL2 GL2 PO4
30 HX GL2 GL2 GL2 PO4
31 HY GL2 GL2 GL2 PO4
32 O31 GL2
33 C31 GL2 GL2 C1A
34 O32 GL2
35 C32 C1A C1A GL2
36 H2X C1A C1A GL2
37 H2Y C1A C1A GL2
38 C23 C1B
39 H3R C1B
40 H3S C1B
41 C24 C1B C1B C2B
42 H4R C1B C1B C2B
43 H4S C1B C1B C2B
44 C25 C1B C2B
45 H5R C1B C2B
46 H5S C1B C2B
47 C26 C2B C2B C1B
48 H6R C2B C2B C1B
49 H6S C2B C2B C1B
50 C27 C2B
51 H7R C2B
52 H7S C2B
53 C28 C2B D3B
54 H8R C2B D3B
55 H8S C2B D3B
56 C29 D3B D3B C2B
57 H91 D3B
58 C210 D3B D3B C4B
59 H101 D3B
60 C211 C4B D3B
61 H11R C4B D3B
62 H11S C4B D3B
63 C212 C4B
64 H12R C4B
65 H12S C4B
66 C213 C4B C4B C5B
67 H13R C4B C4B C5B
68 H13S C4B C4B C5B
69 C214 C4B C4B C5B
70 H14R C4B C4B C5B
71 H14S C4B C4B C5B
72 C215 C5B C4B
73 H15R C5B C4B
74 H15S C5B C4B
75 C216 C5B C5B C4B
76 H16R C5B C5B C4B
77 H16S C5B C5B C4B
78 C217 C5B C5B C5B C4B
79 H17R C5B C5B C5B C4B
80 H17S C5B C5B C5B C4B
81 C218 C5B
82 H18R C5B
83 H18S C5B
84 H18T C5B
85 C33 C1A
86 H3X C1A
87 H3Y C1A
88 C34 C1A C1A C2A
89 H4X C1A C1A C2A
90 H4Y C1A C1A C2A
91 C35 C1A C2A
92 H5X C1A C2A
93 H5Y C1A C2A
94 C36 C2A C2A C1A
95 H6X C2A C2A C1A
96 H6Y C2A C2A C1A
97 C37 C2A
98 H7X C2A
99 H7Y C2A
100 C38 C2A C2A C2A C3A
101 H8X C2A C2A C2A C3A
102 H8Y C2A C2A C2A C3A
103 C39 C2A C2A C3A

```

```

104 H9X C2A C2A C3A
105 H9Y C2A C2A C3A
106 C310 C3A C2A
107 H10X C3A C2A
108 H10Y C3A C2A
109 C311 C3A C3A C2A
110 H11X C3A C3A C2A
111 H11Y C3A C3A C2A
112 C312 C3A
113 H12X C3A
114 H12Y C3A
115 C313 C3A C3A C4A
116 H13X C3A C3A C4A
117 H13Y C3A C3A C4A
118 C314 C4A C3A
119 H14X C4A C3A
120 H14Y C4A C3A
121 C315 C4A C4A C3A
122 H15X C4A C4A C3A
123 H15Y C4A C4A C3A
124 C316 C4A
125 H16X C4A
126 H16Y C4A
127 H16Z C4A

; making
[out]
C13 N C3 C11
[chiral]
H12A C12 N C11 C13
[chiral]
C13 C12 H12A C11 N
O13A C12 H12A C11 N
O13B C12 H12A C11 N

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipind chains
[ out ]
H9R C29 C28 C210
[ trans ]
H10R C210 C29 C28
[ out ]
C211 C210 C29 H10R
__MAP__

# pops.charmm36.map
cat << __MAP__ > pops.charmm36.map
[ molecule ]
POPS

[ martini ]
CNO PO4 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

[ mapping ]
charmm27 charmm36

[ atoms ]
1 N CNO
2 HN1 CNO
3 HN2 CNO
4 HN3 CNO
5 C12 CNO CNO CNO PO4
6 H12A CNO CNO CNO PO4
7 C13 CNO
8 O13A CNO
9 O13B CNO
10 C11 PO4 CNO
11 H11A PO4 CNO
12 H11B PO4 CNO
13 P PO4
14 O13
15 O14
16 O12
17 O11 PO4 PO4 GL1
18 C1 GL1 GL1 PO4
19 HA GL1 GL1 PO4
20 HB GL1 GL1 PO4
21 C2 GL1 GL2
22 HS GL1 GL2
23 O21 GL1 GL1 GL1 C1B
24 C21 GL1 C1B
25 O22 GL1
26 C22 C1B C1B GL1
27 H2R C1B C1B GL1
28 H2S C1B C1B GL1
29 C3 GL2 GL2 GL2 PO4
30 HX GL2 GL2 GL2 PO4
31 HY GL2 GL2 GL2 PO4
32 O31 GL2
33 C31 GL2 GL2 C1A
34 O32 GL2
35 C32 C1A C1A GL2
36 H2X C1A C1A GL2
37 H2Y C1A C1A GL2
38 O23 C1B
39 H3R C1B
40 H3S C1B
41 C24 C1B C1B C2B
42 H4R C1B C1B C2B
43 H4S C1B C1B C2B
44 C25 C1B C2B
45 H5R C1B C2B
46 H5S C1B C2B
47 C26 C2B C2B C1B
48 H6R C2B C2B C1B
49 H6S C2B C2B C1B
50 C27 C2B
51 H7R C2B
52 H7S C2B
53 C28 C2B D3B
54 H8R C2B D3B
55 H8S C2B D3B
56 C29 D3B D3B C2B
57 H9I D3B

```



```

58 C210 D3B D3B C4B
59 H101 D3B
60 C211 C4B D3B
61 H11R C4B D3B
62 H11S C4B D3B
63 C212 C4B
64 H12R C4B
65 H12S C4B
66 C213 C4B C4B C4B C5B
67 H13R C4B C4B C4B C5B
68 H13S C4B C4B C4B C5B
69 C214 C4B C4B C5B
70 H14R C4B C4B C5B
71 H14S C4B C4B C5B
72 C215 C5B C4B
73 H15R C5B C4B
74 H15S C5B C4B
75 C216 C5B C5B C4B
76 H16R C5B C5B C4B
77 H16S C5B C5B C4B
78 C217 C5B C5B C5B C4B
79 H17R C5B C5B C5B C4B
80 H17S C5B C5B C5B C4B
81 C218 C5B
82 H18R C5B
83 H18S C5B
84 H18T C5B
85 C33 C1A
86 H3X C1A
87 H3Y C1A
88 C34 C1A C1A C2A
89 H4X C1A C1A C2A
90 H4Y C1A C1A C2A
91 C35 C1A C2A
92 H5X C1A C2A
93 H5Y C1A C2A
94 C36 C2A C2A C1A
95 H6X C2A C2A C1A
96 H6Y C2A C2A C1A
97 C37 C2A
98 H7X C2A
99 H7Y C2A
100 C38 C2A C2A C2A C3A
101 H8X C2A C2A C2A C3A
102 H8Y C2A C2A C2A C3A
103 C39 C2A C2A C3A
104 H9X C2A C2A C3A
105 H9Y C2A C2A C3A
106 C310 C3A C2A
107 H10X C3A C2A
108 H10Y C3A C2A
109 C311 C3A C3A C2A
110 H11X C3A C3A C2A
111 H11Y C3A C3A C2A
112 C312 C3A
113 H12X C3A
114 H12Y C3A
115 C313 C3A C3A C4A
116 H13X C3A C3A C4A
117 H13Y C3A C3A C4A
118 C314 C4A C3A
119 H14X C4A C3A
120 H14Y C4A C3A
121 C315 C4A C4A C3A
122 H15X C4A C4A C3A
123 H15Y C4A C4A C3A
124 C316 C4A
125 H16X C4A
126 H16Y C4A
127 H16Z C4A

; making
[out]
C13 N C3 C11
[chiral]
H12A C12 N C11 C13
[chiral]
C13 C12 H12A C11 N
O13A C12 H12A C11 N
O13B C12 H12A C11 N

;making R stereoisomer- placing HS
[chiral]
HS C2 O21 C1 C3

; acyl esters
[trans]
C22 C21 O21 C2
[ out ]
O22 C21 O21 C22
[trans]
C32 C31 O31 C3
[out]
O32 C31 O31 C32

;cis bonds in lipind chains
[ out ]
H9R C29 C28 C210
[ trans ]
H10R C210 C29 C28
[ out ]
C211 C210 C29 H10R
__MAP__

# pops.gromos.map
cat << __MAP__ > pops.gromos.map
; Created by Kri on 11.03.2013
[ molecule ]
POPS

[ martini ]
CNO P04 GL1 GL2 C1A C2A C3A C4A C1B C2B D3B C4B C5B

;
; CNO-PO4-GL1-C1B-C2B-D3B-C4B-C5B
; |
; GL2-C1A-C2A-C3A-C4A

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
; Terminal head group (serine)
1 H1 CNO
2 H2 CNO
3 H3 CNO
4 NTM CNO

```

```

5   CA   CNO CNO CNO PO4
6   C6   CNO
7   O7   CNO
8   O8   CNO
9   CE   CNO PO4
; Phosphate group
10  OA   PO4 PO4 PO4 CNO
11  P    PO4
12  OB   PO4
13  OC   PO4
14  OD   PO4 PO4 GL1
; Diacylglycerol
15  CC   GL1 GL1 PO4
16  CD   GL1
17  OE   GL1 GL1 GL1 C1B
18  C1A   GL1 C1B
19  OF
20  C1B   C1B C1B C1B GL1
21  C1C   C1B
22  C1D   C1B C1B C2B
23  C1E   C1B C2B
24  C1F   C2B C2B C1B
25  C1G   C2B
26  C1H   C2B D3B
27  C1I   D3B D3B C2B
28  C1J   D3B D3B C4B
29  C1K   D3B C4B
30  C1L   C4B
31  C1M   C4B C4B C4B C5B
32  C1N   C4B C4B C5B
33  C1O   C5B C4B
34  C1P   C5B C5B C4B
35  C1Q   C5B C5B C5B C4B
36  C1R   C5B
37  CE   GL2
38  OG   GL2 GL2 GL2 C1A
39  C2A   GL2 C1A
40  OH
41  C2B   C1A C1A C1A GL2
42  C2C   C1A
43  C2D   C1A C1A C1A C2A
44  C2E   C1A C2A
45  C2F   C2A C2A C2A C1A
46  C2G   C2A
47  C2H   C2A C2A C2A C3A
48  C2I   C2A C3A
49  C2J   C3A C3A C3A C2A
50  C2K   C3A
51  C2L   C3A C3A C3A C4A
52  C2M   C3A C4A
53  C2N   C4A C4A C3A
54  C2O   C4A C4A C4A C3A
55  C2P   C4A

;;; head group
[chiral]
C6 NTM CE CA
O7 NTM CE CA
O8 NTM CE CA

[ cis ]
;cis double bond
C1H C1I C1J C1K

; Acyl esters
; =====
; This reconstruction is somewhat complex. Unfortunately
; the Gromos united atom force field does not have
; correct dihedrals for acyl esters and these groups
; have to be built with correct geometry. Only setting
; the C-O-CO-C dihedrals correct is not sufficient, as
; the distortions may be so large that the dihedrals
; are pushed over the barrier. Therefore, the whole
; glycerol group is rebuilt so as to form a buffer.

; Acyl ester 1
; -----

[ chiral ]
x CD OE CE CC

[ trans ]
OF C1A CD x

[ out ]
OE C1A OF C1B

[ trans ]
C1B C1A OE CD

[ out ]
OF C1A OE C1B

; Acyl ester 2
; -----

[ out ]
y CE CD OG

[ chiral ]
z CE OG CD y

[ trans ]
OH C2A CE z

[ out ]
OG C2A OH C2B

[ trans ]
C2B C2A OG CE

[ out ]
OH C2A OG C2B
__MAP__

# pro.amber.map
cat << __MAP__ > pro.amber.map
[ molecule ]
PRO

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1      N      BB

```

```

2    CD    SC1
3    HD1   SC1
4    HD2   SC1
5    CG    SC1
6    HG1   SC1
7    HG2   SC1
8    CB    SC1 BB
9    HB1   SC1 BB
10   HB2   SC1 BB
11   CA    BB
12   HA    BB
13   C     BB
14   O     BB

[ chiral ]
CB    CA    N    C
HB1   CA    N    C
HB2   CA    N    C

[ chiral ]
HA    CA    N    CB    C ; L-Pro
; HA    CA    N    C    CB ; D-Pro
__MAP__

# pro.charmm36.map
cat << __MAP__ > pro.charmm36.map
[ molecule ]
PRO

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
1    N     BB
2    CD    SC1
3    HD1   SC1
4    HD2   SC1
5    CA    BB
6    HA    BB
7    CB    SC1 BB
8    HB1   SC1 BB
9    HB2   SC1 BB
10   CG    SC1
11   HG1   SC1
12   HG2   SC1
13   C     BB
14   O     BB

[ chiral ]
CB    CA    N    C
HB1   CA    N    C
HB2   CA    N    C

[ chiral ]
HA    CA    N    CB    C ; L-Pro
; HA    CA    N    C    CB ; D-Pro
__MAP__

# pro.gromos.map
cat << __MAP__ > pro.gromos.map
[ molecule ]
PRO

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1    N     BB
2    CA    BB
3    CB    SC1 BB
4    CG    SC1
5    CD    SC1
6    C     BB
7    O     BB

[ chiral ]
CB    CA    N    C
__MAP__

# sep.gromos.map
cat << __MAP__ > sep.gromos.map
[ molecule ]
SEP

[ martini ]
BB SC1 PO4

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1    N     BB
2    H     BB
3    CA    BB
4    CB    SC1 BB
5    OG    SC1
6    P     PO4
7    O1P
8    O2P
9    O3P
10   C     BB
11   O     BB

[ chiral ]
CB    CA    N    C

[ out ]
O2P P O1P OG
[ chiral ]
O3P P O2P OG
O1P P O2P O3P OG
__MAP__

# ser.amber.map
cat << __MAP__ > ser.amber.map
[ molecule ]
SER

[ martini ]

```

```

BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      OG      SC1 SC1 BB
  9      HG      SC1
 10      C       BB
 11      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Ser
; HA     CA      N      C      CB ; D-Ser
__MAP__

# ser.charmm36.map
cat << __MAP__ > ser.charmm36.map
[ molecule ]
SER

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
  1      N      BB
  2      HN      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB1     SC1 BB BB
  7      HB2     SC1 BB BB
  8      OG      SC1 SC1 BB
  9      HG1     SC1
 10      C       BB
 11      O       BB

[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Ser
; HA     CA      N      C      CB ; D-Ser
__MAP__

# ser.gromos.map
cat << __MAP__ > ser.gromos.map
[ molecule ]
SER

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      CB      SC1 BB BB
  5      OG      SC1 SC1 BB
  6      HG      SC1
  7      C       BB
  8      O       BB

[ chiral ]
CB      CA      N      C
__MAP__

# sol.gromos.map
cat << __MAP__ > sol.gromos.map
[ molecule ]
SOL

[ martini ]
__MAP__

# thr.amber.map
cat << __MAP__ > thr.amber.map
[ molecule ]
THR

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1      N      BB
  2      H      BB
  3      CA      BB
  4      HA      BB
  5      CB      SC1 BB BB
  6      HB      SC1 BB BB
  7      CG2     SC1
  8      HG21    SC1
  9      HG22    SC1
 10      HG23    SC1
 11      OG1     SC1 SC1 BB
 12      HG1     SC1
 13      C       BB
 14      O       BB

[ chiral ]
CB      CA      N      C
HB      CA      N      C

```

```

[ chiral ]
  HA   CA   N   CB   C ; L-Thr
; HA   CA   N   C   CB ; D-Thr

[ out ]
OG1 CB CG2 CA
[ trans ]
HG1 OG1 CB CA

[ chiral ]
  HB   CB   CG2   OG1 CA ; 3R stereoisomer (natural form)
; HB   CB   CG2   CA   CG1 ; 3S stereoisomer
__MAP__

# thr.charmm36.map
cat << __MAP__ > thr.charmm36.map
[ molecule ]
THR

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
  1   N   BB
  2   HN  BB
  3   CA   BB
  4   HA   BB
  5   CB   SC1 BB BB
  6   HB   SC1 BB BB
  7   OG1  SC1 SC1 BB
  8   HG1  SC1
  9   CG2  SC1
 10  HG21  SC1
 11  HG22  SC1
 12  HG23  SC1
 13   C   BB
 14   O   BB

[ chiral ]
  CB   CA   N   C
  HB   CA   N   C

[ chiral ]
  HA   CA   N   CB   C ; L-Thr
; HA   CA   N   C   CB ; D-Thr

[ out ]
OG1 CB CG2 CA
[ trans ]
HG1 OG1 CB CA

[ chiral ]
  HB   CB   CG2   OG1 CA ; 3R stereoisomer (natural form)
; HB   CB   CG2   CA   CG1 ; 3S stereoisomer
__MAP__

# thr.gromos.map
cat << __MAP__ > thr.gromos.map
[ molecule ]
THR

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
  1   N   BB
  2   H   BB
  3   CA   BB
  4   CB   SC1 BB BB
  5   OG1  SC1 SC1 BB
  6   HG1  SC1
  7   CG2  SC1 SC1 BB
  8   C   BB
  9   O   BB

[ chiral ]
  CB   CA   N   C
__MAP__

# trp.amber.map
cat << __MAP__ > trp.amber.map
[ molecule ]
TRP

[ martini ]
BB SC1 SC2 SC3 SC4

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
  1   N   BB
  2   H   BB
  3   CA   BB
  4   HA   BB
  5   CB   SC1 BB
  6   HB1  SC1 BB
  7   HB2  SC1 BB
  8   CG   SC1 SC1 SC1 SC1 SC3 SC3
  9   CD1  SC1
 10  HD1  SC1
 11  NE1  SC2 SC1
 12  HE1  SC2
 13  CE2  SC2 SC2 SC3
 14  C22  SC2 SC2 SC4
 15  H22  SC2
 16  CH2  SC4 SC4 SC2
 17  HH2  SC4
 18  C23  SC4 SC4 SC3
 19  H23  SC4
 20  CE3  SC3 SC3 SC4
 21  HE3  SC3
 22  CD2  SC3 SC3 SC2
 23   C   BB
 24   O   BB

[ trans ]
CB CG CD2 CE2
HD1 CD1 NE1 CE2

```

```

HE1 NE1 CD1 CG
HE3 CE3 CD2 CE2
H22 CZ2 CE2 CD2
H23 CZ3 CE3 CD2
HH2 CH2 CZ3 CE3

[ chiral ]
CB   CA   N   C
HB1  CA   N   C
HB2  CA   N   C

[ chiral ]
HA   CA   N   CB   C ; L-Trp
; HA   CA   N   C   CB ; D-Trp
__MAP__

# trp.charmm36.map
cat << __MAP__ > trp.charmm36.map
[ molecule ]
TRP

[ martini ]
BB SC1 SC2 SC3 SC4

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN     BB
3      CA     BB
4      HA     BB
5      CB     SC1 BB
6      HB1    SC1 BB
7      HB2    SC1 BB
8      CG     SC1 SC1 SC1 SC1 SC3 SC3
9      CD1    SC1
10     HD1    SC1
11     NE1    SC2 SC1
12     HE1    SC2
13     CE2    SC2 SC2 SC3
14     CD2    SC3 SC3 SC2
15     CE3    SC3 SC3 SC4
16     HE3    SC3
17     CZ3    SC4 SC4 SC3
18     H23    SC4
19     CZ2    SC2 SC2 SC4
20     H22    SC2
21     CH2    SC4 SC4 SC2
22     HH2    SC4
23      C     BB
24      O     BB

[ trans ]
CB   CG   CD2   CE2
HD1  CD1  NE1   CE2
HE1  NE1  CD1   CG
HE3  CE3  CD2   CE2
H22  CZ2  CE2   CD2
H23  CZ3  CE3   CD2
HH2  CH2  CZ3   CE3

[ chiral ]
CB   CA   N   C
HB1  CA   N   C
HB2  CA   N   C

[ chiral ]
HA   CA   N   CB   C ; L-Trp
; HA   CA   N   C   CB ; D-Trp
__MAP__

# trp.gromos.map
cat << __MAP__ > trp.gromos.map
[ molecule ]
TRP

[ martini ]
BB SC1 SC2 SC3 SC4

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA     BB
4      CB     SC1 BB
5      CG     SC1 SC1 SC1 SC1 SC3 SC3
6      CD1    SC1
7      HD1    SC1
8      CD2    SC3 SC3 SC2
9      NE1    SC2 SC1
10     HE1    SC2
11     CE2    SC2 SC2 SC3
12     CE3    SC3 SC3 SC4
13     HE3    SC3
14     CZ2    SC2 SC2 SC4
15     H22    SC2
16     CZ3    SC4 SC4 SC3
17     H23    SC4
18     CH2    SC4 SC4 SC2
19     HH2    SC4
20      C     BB
21      O     BB

[ trans ]
CB   CG   CD2   CE2
HD1  CD1  NE1   CE2
HE1  NE1  CD1   CG
HE3  CE3  CD2   CE2
H22  CZ2  CE2   CD2
H23  CZ3  CE3   CD2
HH2  CH2  CZ3   CE3

[ chiral ]
CB   CA   N   C
__MAP__

# tyr.amber.map
cat << __MAP__ > tyr.amber.map
[ molecule ]
TYR

[ martini ]
BB SC1 SC2 SC3

```

```
[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS
```

```
[ atoms ]
 1      N      BB
 2      H      BB
 3      CA     BB
 4      HA     BB
 5      CB     SC1 BB
 6      HB1    SC1 BB
 7      HB2    SC1 BB
 8      CG     SC1
 9      CD1    SC1 SC3
10      HD1    SC1
11      CE1    SC3
12      HE1    SC3
13      CZ     SC3 SC3 SC1 SC2 SC2
14      OH     SC3 SC3 SC2
15      HH     SC3
16      CE2    SC2
17      HE2    SC2
18      CD2    SC2 SC1
19      HD2    SC2
20      C      BB
21      O      BB
```

```
[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C
```

```
[ chiral ]
HA      CA      N      CB      C ; L-Tyr
; HA     CA      N      C      CB ; D-Tyr
```

```
[ out ]
; Add some helper points
P CD1 CD2 CZ
Q CD2 CD1 CZ
R CZ   CD1 CD2
```

```
[ trans ]
; Place hydrogens using helper points
HD1  P CE1  R
HD2  Q CE2  R
HE1  CE1  P CG
HE2  CE2  Q CG
OH   R CE1  P
HH   OH   R CE1
```

```
[ out ]
; Place ring carbons
CD1 CE1 HE1 R
CD2 CE2 HE2 R
CZ  CE1 HE1 P
__MAP__
```

```
# tyr.charmm36.map
cat << __MAP__ > tyr.charmm36.map
[ molecule ]
TYR
```

```
[ martini ]
BB SC1 SC2 SC3
```

```
[ mapping ]
charmm27 charmm36
```

```
[ atoms ]
 1      N      BB
 2      HN     BB
 3      CA     BB
 4      HA     BB
 5      CB     SC1 BB
 6      HB1    SC1 BB
 7      HB2    SC1 BB
 8      CG     SC1
 9      CD1    SC1 SC3
10      HD1    SC1
11      CE1    SC3
12      HE1    SC3
13      CZ     SC3 SC3 SC1 SC2 SC2
14      OH     SC3 SC3 SC2
15      HH     SC3
16      CD2    SC2 SC1
17      HD2    SC2
18      CE2    SC2
19      HE2    SC2
20      C      BB
21      O      BB
```

```
[ chiral ]
CB      CA      N      C
HB1     CA      N      C
HB2     CA      N      C
```

```
[ chiral ]
HA      CA      N      CB      C ; L-Tyr
; HA     CA      N      C      CB ; D-Tyr
```

```
[ out ]
; Add some helper points
P CD1 CD2 CZ
Q CD2 CD1 CZ
R CZ   CD1 CD2
```

```
[ trans ]
; Place hydrogens using helper points
HD1  P CE1  R
HD2  Q CE2  R
HE1  CE1  P CG
HE2  CE2  Q CG
OH   R CE1  P
HH   OH   R CE1
```

```
[ out ]
; Place ring carbons
CD1 CE1 HE1 R
CD2 CE2 HE2 R
CZ  CE1 HE1 P
__MAP__
```

```
# tyr.gromos.map
cat << __MAP__ > tyr.gromos.map
[ molecule ]
TYR
```

```

[ martini ]
BB SC1 SC2 SC3

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA      BB
4      CB      SC1 BB
5      CG      SC1
6      CD1     SC1 SC3
7      HD1     SC1
8      CD2     SC2 SC1
9      HD2     SC2
10     CE1     SC3
11     HE1     SC3
12     CE2     SC2
13     HE2     SC2
14     CZ      SC3 SC3 SC1 SC2 SC2
15     OH      SC3 SC3 SC2
16     HH      SC3
17      C      BB
18     O      BB

[ chiral ]
CB      CA      N      C

[ out ]
; Add some helper points
P CD1 CD2 CZ
Q CD2 CD1 CZ
R CZ   CD1 CD2

[ trans ]
; Place hydrogens using helper points
HD1  P CE1  R
HD2  Q CE2  R
HE1  CE1  F  CG
HE2  CE2  Q  CG
OH   R CE1  P
HH   OH  R CE1

[ out ]
; Place ring carbons
CD1 CE1 HE1 R
CD2 CE2 HE2 R
CZ  CE1 HE1 P
__MAP__

# val.amber.map
cat << __MAP__ > val.amber.map
[ molecule ]
VAL

[ martini ]
BB SC1

[ mapping ]
amber amber94 amber96 amber99 amber99sb amber03 amberGS

[ atoms ]
1      N      BB
2      H      BB
3      CA      BB
4      HA      BB
5      CB      SC1 BB
6      HB      SC1 BB
8      CG1     SC1
9      HG11    SC1
10     HG12    SC1
11     HG13    SC1
12     CG2     SC1
13     HG21    SC1
14     HG22    SC1
15     HG23    SC1
16      C      BB
17     O      BB

[ chiral ]
CB      CA      N      C
HB      CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Val
; HA      CA      N      C      CB ; D-Val

[ out ]
CG2 CB CG1 CA
HG21 CB CG1 CA
HG22 CB CG1 CA
HG23 CB CG1 CA
__MAP__

# val.charmm36.map
cat << __MAP__ > val.charmm36.map
[ molecule ]
VAL

[ martini ]
BB SC1

[ mapping ]
charmm27 charmm36

[ atoms ]
1      N      BB
2      HN      BB
3      CA      BB
4      HA      BB
5      CB      SC1 BB
6      HB      SC1 BB
8      CG1     SC1
9      HG11    SC1
10     HG12    SC1
11     HG13    SC1
12     CG2     SC1
13     HG21    SC1
14     HG22    SC1
15     HG23    SC1
16      C      BB
17     O      BB

[ chiral ]
CB      CA      N      C

```



```

      HB      CA      N      C

[ chiral ]
HA      CA      N      CB      C ; L-Val
; HA      CA      N      C      CB ; D-Val

[ out ]
CG2 CB CG1 CA
HG21 CB CG1 CA
HG22 CB CG1 CA
HG23 CB CG1 CA
__MAP__

# val.gromos.map
cat << __MAP__ > val.gromos.map
[ molecule ]
VAL

[ martini ]
BB SC1

[ mapping ]
gromos gromos43a1 gromos43a2 gromos45a3 gromos53a5 gromos53a6 gromos54a7

[ atoms ]
1      N      BB
2      H      BB
3      CA      BB
4      CB      SC1 BB
5      CG1      SC1
6      CG2      SC1
7      C      BB
8      O      BB

[ chiral ]
CB      CA      N      C
__MAP__

```